UCRL-WEB-200124

EZJOBCONTROL

Table of Contents

Preface

Scope:	EZJOBCONTROL supplements reference material on the LC batch system by introducing and comparing the basic issues, tools, techniques, and alternatives for job planning and execution on LC machines, including the LCRM (formerly DPCS) batch-processing system. EZJOBCONTROL aims to help you build, run, and monitor (batch) jobs more effectively, while the LCRM (DPCS) Reference Manual (URL: http://www.llnl.gov/LCdocs/dpcs) aims to reveal and explain LC's across-machine scheduling and job-control (metabatch) system and the Bank and Allocation Manual (URL: http://www.llnl.gov/LCdocs/banks) explains LC's time management policies. The reference manuals take the comprehensive, system perspective while EZJOBCONTROL takes the task-oriented, user perspective to job management topics, tools, and common batch-job problems.
Availability:	When the programs described here are limited by machine, those limits are included in their explanation. Otherwise, they run under any LC UNIX system.
Consultant:	For help contact the LC customer service and support hotline at 925-422-4531 (open e-mail: lc-hotline@llnl.gov, SCF e-mail: lc-hotline@pop.llnl.gov).
Printing:	The print file for this document can be found at:

```
OCF: http://www.llnl.gov/LCdocs/ezjob/ezjob.pdf
SCF: http://www.llnl.gov/LCdocs/ezjob/ezjob_scf.pdf
```

Introduction

EZJOBCONTROL introduces the alternative ways to run programs and manage jobs on the LC machines, and it explains the basic features of the local batch system, called the Livermore Computing Resource Management (LCRM) system (but long known by its former name of Distributed Production Control System (DPCS)). This is a start-quickly user guide, not a comprehensive reference manual (which is a separate document (URL: http://www.llnl.gov/LCdocs/dpcs)).

This guide summarizes the LCRM job-scheduling policies and algorithms. It includes a typical batch script thoroughly annotated to explain the role of each feature, as well as a list of script features that support special environments (such as IBM's POE (URL: http://www.llnl.gov/LCdocs/poe)) or that will not transfer between different LC machines. Step-by-step instructions show how to plan, run, monitor, and change the specifications of a typical batch job successfully. Also included is a discussion of how several common mistakes are handled by LCRM or the operating system.

Scheduling.
LC uses a very dynamic "fair-share" approach to political (competitive) scheduling of jobs, instead of a more traditional job-scheduling approach based only on job features and fixed computer-time allocations. EZJOBCONTROL introduces the general software tools that monitor and manage your computing "shares," and it suggests plausible ways to plan your job and handle its constraints in light of this fair-share scheduling. If you are one of the few LC users authorized to manage resource banks (page 8), consult the Bank and Allocation Manual (URL: http://www.llnl.gov/LCdocs/banks) for details on extra bank-management and resource-montitoring tools and for specific instructions on how to change user assignments and time allocations. LCRM completely eliminated the former "account" attribute of batch jobs in August, 2005.

Resource Management.
Starting in 2003, the former DPCS began changing its official name to the Livermore Computing Resource Management (LCRM) system. This change chiefly affects internal files, features, and libraries, however. The job-control user utilities (discussed here) and most user messages from LCRM/DPCS remain unchanged. Also starting in 2003, LC began deploying on all of its production Linux (CHAOS) machines a locally designed, low-level resource manager to work "below" LCRM/DPCS (from a user's viewpoint) to more efficiently handle nodes and tasks for large parallel jobs. See the SLURM Reference Manual (URL: http://www.llnl.gov/LCdocs/slurm) for details on what this low-level system contributes to within-machine job control and resource allocation in the LC Linux environment.

Moab Role.
Starting in March, 2007, LC began gradually replacing LCRM (DPCS) with a commercial product called "Moab Workload Manager" (MWM, or simply Moab) from Cluster Resources, Inc. To implement this replacement, LC automatically translates some LCRM features and tool options into Moab counterparts, but it modifies or abandons others. EZJOBCONTROL now mentions specific cases of this LCRM-to-Moab replacement throughout the text. For a more comprehensive look at Moab's impact on LC production computing, however, see Moab at LC (URL: http://www.llnl.gov/LCdocs/moab).

Alternatives for Running Programs

There are many alternative ways to execute a job on LC machines:

```
Interactively
     In the foreground (default)
     In the background (& suffix)
          Ending at logout (default)
          Continuing after logout (NOHUP)

Using local Linux (CHAOS) batch queues (via SRUN)

Using across-machine batch scheduling (via LCRM or Moab)
     Without terminal access (default)
     With terminal access via RUN/PROXY
```

Interactive Jobs.
In general, all LC production machines limit *interactive* jobs to no more than 30 CPU minutes/process and no more than 1/10 the physical memory of the machine where they run. (If you interactively execute one script that invokes several processes, each separate process is allowed 30 CPU minutes. However, if several processes associated with the same "job" script really consume as much as 30 min each, the LC User Services staff will detect this and contact you to urge using batch execution instead in the future.)

Batch Jobs.
Details may vary, but the goal of such interactive limits is always to encourage you to run jobs that are long, large, or both in the batch system, for which the "Livermore Computing Resource Management" system (LCRM, formerly DPCS) provides the controlling utilities at LC. (Sometimes, LC temporarily allows long interactive runs on a new machine if LCRM is not yet installed there.) Because batch jobs call for extra preparation and planning, while offering you in exchange much more size and scope flexibility, most sections in this guide discuss job control from the batch perspective.

As Moab gradually replaces LCRM on LC production machines, this strategy for managing batch jobs (and even the familiar job-scheduling priorities) remains unchanged, but some new tools and tool options take the place of traditional LCRM job-control utilities (they are noted below in appropriate places in the text).

On LC's Linux (CHAOS) machines, the SRUN utility (URL: http://www.llnl.gov/LCdocs/slurm/index.jsp?show=s4.2) can either launch your job from within an LCRM-managed script for across-machine scheduling and accounting, or else manage your job's resources in a "local (one-machine) batch" mode even when LCRM is unavailable (see the SLURM Reference Manual (URL: http://www.llnl.gov/LCdocs/slurm) for examples).

The next section compares interactive and batch jobs in terms of LC's job-scheduling policy. The last section (page 47) in this guide returns to this comparison again, in terms of job problems and troubleshooting.

Access to LCRM Utilities.

Later sections of this guide introduce and explain the user utility programs (PSUB, PSTAT, PALTER, and others) by which you interact with LC's batch system (to submit and track your jobs). For efficiency, LC now makes these LCRM software tools available *only* on:
(1) the few nodes of each cluster designated as interactive login nodes, and
(2) the single node on which any single-node job executes, and
(3) the master node (but *not* the other nodes) on which a multiple-node job executes.

Access to Job Nodes.

Besides a few designated login nodes on every LC cluster, users running a batch job are also allowed to log in to any compute node on which that job executes, usually to run additional monitoring or job-guiding interactive processes.
(1) Remember that the LCRM utilities may not be available on compute nodes, as noted above.
(2) Such access to a job's compute nodes is only allowed (on SLURM-managed machines) *after* the job's first execution of POE or MPIRUN (which sometimes causes a noticeable delay in login access).
(3) Starting in August, 2007, on LC Linux (but *not* AIX) clusters where SLURM is the resource manager underlying LCRM or Moab, *all* of a user's processes on any compute node will terminate whenever a SLURM-managed batch job completes on that node. This guarantees that the next user's job will see no interference from stray, CPU-intensive processes that accompanied your job. If your login session or interactive process on a Linux compute node unexpectedly ends, your batch job to which that node was allocated has probably just completed.

Job-Scheduling Policy

Policy Background

For job-scheduling and resource-assignment purposes, the difference between interactive and batch jobs at LC is much less than at many other UNIX sites. For example, on LC machines there is

- no difference in NICE (run priority) value between foreground and background jobs,
- no NICE-control option for the PSUB batch-submittal utility at all, and
- no overt job-scheduling queue structure differentiated by explicit NICE values (and hence by relative delivery rates of time to jobs) or memory limits.

Instead, resource managers can specify maximum values on CPU time or memory usage for interactive and for batch jobs alike (for example, some YANA nodes run no batch jobs, some run 12-hour batch jobs, and some run 200-hour jobs). Whole machines can be tuned to favor interactive sessions or else batch production runs, a very coarse-grained and nonexclusive way to sort jobs by type. For each separate machine on which LCRM schedules jobs, the system checks submitted jobs every 20 sec to detect jobs precluded from running next (usually because of constraints like those just mentioned). Then it picks the best candidate to run next (from the remaining candidate job pool), using a complex scheduling algorithm described in the LCRM (DPCS) Reference Manual (URL: http://www.llnl.gov/LCdocs/dpcs/index.jsp?show=s4.5.2).

"Scheduling" is itself ambiguous, and has at least three meanings at LC:

- LOAD BALANCING--dividing jobs among available machines or processors efficiently. SLURM, for example, can invoke any of three different schedulers (URL: http://www.llnl.gov/LCdocs/slurm/index.jsp?show=s3.4) to spread jobs among compute nodes in different ways.
- GANG SCHEDULING--grouping the related, coordinated tasks of a parallel program so they can all get processors at the same time and can time-share those processors with other jobs effectively. Currently at LC, you must overtly (prepare for and) request gang scheduling of parallel jobs, using techniques explained in the Gang Scheduler User Guide. (URL: http://www.llnl.gov/LCdocs/gang)
- POLITICAL SCHEDULING--spreading compute resources "properly" among users or groups of users. In fact, allocating CPU time to users and deciding when their jobs will run, which are conceptually distinct problems, merge entirely at LC under the "fair-share scheduling" approach now used on all open and secure production machines.

The dominant political scheduling mechanism at LC is not placement of individual jobs in explicit, ranked job queues. Instead, scheduling is driven chiefly by (1) the implicit relationship of a job's "bank" to other banks, and (2) the "shares" and past usage profile of the job's owner (relative to other owners). The next two sections explain these indirect but important scheduling factors.

How Banks Affect Job Scheduling

At LC a bank, which was formerly a pool of CPU time, is now a pool of compute "shares" or unitless entitlements to use computing resources. (LCRM completely eliminated the former "account" attribute of batch jobs in August, 2005.) Every user and every job are associated with some specific bank (by default or choice), and that bank's influence is extensive:

Hierarchy

The parent-child or the sibling relationships among banks influence how the jobs funded from those banks are scheduled:

- Built into LC's formula for "normalizing" compute shares and usage (so they can be compared among users) is an overt multiplicative "parent factor" that scales the result in proportion to your subbank's fraction of its parent bank's shares.
- Normalization applies recrusively through the hierarchy of ALL banks. So all parents of the bank associated with your job always affect the current normalized value of your shares, and thereby your job's fair-share priority.

Resource Sharing

To allow ad hoc adjustments to the general allocation (and hence scheduling) "flow" described above, any specific bank can be...

- Declared a resource-sharing bank, enabling shares to flow to its children (and to the jobs they fund), or
- Excluded from resource sharing even if its sibling banks do share their parent's shares.

Special Cases

With special prior authorization (only), users can declare individual jobs as either "exempted" or "expedited," which schedules that job more quickly by ignoring some or all of the foregoing scheduling algorithms. There are significant limitations on doing this, however, as explained in the "Plan Your Job Constraints" (page 20) section below and in the "Expediting and Exempting Jobs" section of the LCRM (DPCS) Reference Manual (URL: http://www.llnl.gov/LCdocs/dpcs).

Resource Quotas

Resource managers can superimpose on this whole scheme time "quotas" for specified periods (daily, weekly, etc.) such that any bank that exceeds its quota for a period (or that has a parent bank that exceeds the parent quota) will stop funding jobs until the next quota period begins. Note that starting in February, 2006, the former distinction between batch (LCRM-only) and interactive banks ended, so that all banks now serve *both* batch and interactive jobs.

How Shares Affect Job Scheduling

The fair-share priority of every job you own (in the same LCRM partition) is the same, regardless of the job's detailed features. At LC, this fair-share priority reflects almost entirely just the DIFFERENCE between

- your normalized shares (how entitled you are to compute), and
- your normalized usage, where usage has an historical component that takes account of your recent computing activity level over time.

How these two values (normalized shares and usage) are calculated before subtracting the second from the first defies simple description, however. To appreciate the intricacy of the calculation and the degree of indirection involved requires a look at the LC fair-share algorithms, to which a full section in the LCRM (DPCS) Reference Manual (URL: http://www.llnl.gov/LCdocs/dpcs) is devoted. (A term added to the fair-share algorithm in January, 2001, allows the RATIO of normalized shares to normalized usage to influence, or even dominate, the difference term. The weight of this ratio term is initially set to 0, however.)

In practice, you can simply monitor the current results of these (often changing) fair-share calculations by running the PSHARE (page 26) utility, using the options suggested in a later section. PSHARE reports the current normalized shares, normalized usage, and (optionally) resulting job priority for yourself, your competing users, and your bank and its parents. These values may (or may not) help you predict how any job of yours will compete for resources against jobs owned by other users when LCRM tries to schedule them. (Authorized managers only can force a job's priority to a specified value by using the privileged -p option of the PALTER (page 45) utility.) Also, starting in February, 2006, a "full report" on a batch job using PSTAT's -f option includes the current value of a field called "resources used," but which is actually the job's fair-share "aggregate resource units" (AGUs). This is LCRM's fair-share usage estimate, and it allows you to compare the usage of different jobs run in different circumstances.

On LC production machines where Moab has replaced LCRM as the across-cluster batch-job scheduler

```
mdiag -f
```

has replaced PSHARE as the tool for reporting detailed share-like usage information.

Annotated Typical Batch Script

This section contains a simple but typical C-shell script designed to execute typical tasks successfully on LC computers (such as Thunder or UP). Just below each script line is an explanation of its role. These comments introduce batch system default behavior for a variety of circumstances, and then explain how the script modifies that behavior (often) to yield more convenient or more reliable results.

Not every job-control script needs every feature shown here, but omitting features without understanding their role may prevent your batch job from running successfully. Also, the features shown here work well across all LC machines. Other helpful features that are limited to specific types of machines are listed and explained in the next section (page 15), where their discrepant behavior is pointed out so you can plan carefully if you choose to use them.

The maximum length of an LCRM job name (and hence a script name) is 15 characters, while the maximum user name is 31 characters.

When LCRM accepts your submitted job (by returning a job id number) it copies your script (into a "spool" area). Changes you make to your copy after this point will NOT affect how the (original spooled) script executes. So the time to correct flaws in your script is before you submit it to LCRM, not afterward.

(The purpose of this sample script is actually to read a file from storage, process it with the UNIX SPELL utility, and then save the resulting output file in storage, to illustrate a fairly common pattern of job steps.)

SCRIPT LINE	EXPLANATORY COMMENTS
	Imbedded PSUB options must: • begin with the string #PSUB in uppercase. • come before any other script commands except for the choice of shell.
#!/bin/csh	DEFAULT: The default shell may vary between machines and users (but it is often the Bourne shell sh). SUGGESTION: At the very start of your script, use the unique syntax shown here to explicitly declare which shell your job should run under (here, the C shell csh). WARNING: LCRM accepts exactly four shells--sh, csh, ksh, and perl.
#PSUB -eo	DEFAULT: Error messages and log messages go to separate files (called by default *input*.e*jid* and *input*.o*jid*, if your batch script is called *input* and *jid* is your numerical job ID). SUGGESTION: Use PSUB's -eo option to collect all output messages in one file for easier analysis. (See also SET ECHO below.)

#PSUB -o /g/g16/jfk/mylogs/mylog1

DEFAULT: By default the batch system deposits your log file in the online (not storage) directory from which you submitted the job, which is NOT necessarily helpful later when you want to find it.

SUGGESTION: To reliably deposit the log file where you want with a name that you recognize, use PSUB's -o option to specify the absolute pathname for the log file (all directories used here must exist before you run your job, so using an already created child of your common home directory (/g/g16/jfk/mylogs in the example) is a good strategy on any production machine). See also -ro below.

#PSUB -nr DEFAULT: By default the batch system tries to rerun from the start any running job whenever a checkpoint restart fails.
SUGGESTION: You must overtly request -nr (no rerun) if you want it, as is done here, to avoid the likely waste of resources.

#PSUB -ro DEFAULT: By default the batch system automatically holds your job's standard output (error and log messages) in a spool file until your job ends, where it could be lost.
SUGGESTION: You must overtly request -ro (write standard output directly to your specified output file), as is done here.
WARNING: Before you submit your batch script to LCRM you must previously create (with MKDIR) every directory and subdirectory needed to support your standard output file(s). Failure to create these directories in advance causes your job to die as soon as LCRM encounters your PSUB -ro option (you cannot have the job create its own error/log directories because the imbedded PSUB -ro option must come BEFORE, not after, all your ordinary script commands).
A common mistake when using -ro is to create needed error/log directories on only one node in a cluster, but in a file system such as /usr/tmp that is local to each node. If you then manyow the job to run on several (or many) nodes in the cluster and if LCRM pick some for execution that lack the needed directories, your job dies almost immediately (as soon as -ro is encountered). To avoid this problem when you run in any massively parallel environment at LC, create the needed error/log directories in your common home directory (e.g., such as /g/g16/jfk/mylogs), which is automatically available to every machine node. (You could also restrict the job to run only on one or more specific nodes in a cluster by using PSUB's -c option (page 31).)

#PSUB -tM *mm*

DEFAULT: PSUB's -lt option specifies a per-process time limit (in minutes), while option -tM (shown here) specifies a total maximum time limit without imposing time constraints on individual processes within the job.
SUGGESTION: Use either -lt or -tM (or both if you wish) to impose the kind of time limit that best suits your job. Use the format *hh:mm* to specify hours instead of just minutes (so for 2.5 hours use either -tM 150 or -tM 2:30). A separate -tW option limits wall-clock time.
WARNING: The default total-job time limit for batch jobs on LC machines (if you omit -tM) is only 30 minutes. Jobs that exceed their total time limit are killed. To change a mistaken time limit (with the PALTER utility) after you have already submitted your job, see the "Alter Job Features" (page 45) section below.

```
#job commands start here
```

Aside from PSUB imbedded options, all lines that begin with # are just comments (like this one).

```
set echo
```

DEFAULT: On LC machines, no executed commands are automatically echoed in your log file.
SUGGESTION: You must overtly request command echoing if you want it, as is done here, for easier debugging. Strangely, echoed commands appear in your error file unless you use the -eo option (explained above) to combine your error and output files.

```
echo LCRM job id = $PSUB_JOBID
```

DEFAULT: The PSTAT monitoring utility reports only on waiting or running jobs, unless with -T you already know the job's job ID. So you will not be able to discover the job ID of a completed job once it ends, a debugging obstacle if you run several jobs.
SUGGESTION: Include this request to echo into your log file the value of the LCRM environment variable PSUB_JOBID (all uppercase) at the start of every job. The output will be the 5-digit number that uniquely identified your job to LCRM while it ran. (LCRM does include your job ID in all its e-mail to you.)

```
mkdir -p /var/tmp/jfk/myruns
cd /var/tmp/jfk/myruns
```

DEFAULT: All batch jobs start executing in your login directory.
SUGGESTION: Immediately make (MKDIR) a work directory for each job (here /jfk/myruns, a child of standard temp directory /var/tmp) and move (CD) into it. This shelters your job from others and improves run efficiency. Having your job create its own work directory guarantees that the directory will be present regardless of which node runs the job. Using MKDIR's -p option lets you create several directory layers at once, and also avoids an error message if the work directory already exists (from a previous run). Your system administrator MAY have specified a dedicated temporary directory automatically created when your job starts and automatically purged when your job ends. In general, environment variable PCS_TMPDIR contains the location of that directory in case you want to use it, but on LC IBM machines an even better approach is to use LOADL_STEP_ID as explained below in the "Script Differences" section (page 15). (Using a globally mounted /nfs/tmp*n* directory instead of the local /var/tmp will let you access your output from any production node.)

```
ftp storage <<EOF
jfk
cd testdir
get test001
quit
EOF
```

DEFAULT: Files left online may be purged or otherwise lost before your batch job that needs them starts to run.
SUGGESTION: Retrieve all needed data (here, TEST001) or private executable files from storage at the start of your job (here using passwordless FTP by file owner JFK and retrieving TEST001 from storage directory TESTDIR). To be sure that retrieved executables (or scripts created as your job runs) will execute, (re)confirm their executable status by running

```
chmod u+x progname
```

```
/usr/bin/spell test001 >! sp.out
```

DEFAULT: Your current (batch) directory may not be in your search path (or public files with conflicting names may precede it).
SUGGESTIONS:
(1) To run programs not local to your work subdirectory, use their absolute pathnames (as shown here).
(2) To run any local program or script with the same name as any public file in any search-path directory, overtly use its relative pathname, for example

```
./spell
```

(3) Whenever you redirect output to a disk file, always redirect WITH OVERWRITE (e.g., use >! rather than just > in the C shell). Otherwise, leftover previous versions of output files could prevent rerunning your job successfully after a first attempt.

```
ftp storage <<EOF
jfk
cd outdir
put sp.out
quit
EOF
```

DEFAULT: No work (batch output) files are stored or backed up automatically, and they can be purged, even from /nfs/tmp*n* global work directories.
SUGGESTION: To avoid losing your batch output during a file purge or disk crash, place all important output files (here illustrated by SP.OUT) in archival storage before ending your batch job (here, using passwordless FTP to storage directory OUTDIR of job owner JFK).

rm test001	DEFAULT: The batch system does not automatically clean up temporary files, or any files, after a batch job ends. (Days later, a general file-system purge may eventually delete left-over files from /var/tmp.) SUGGESTION: Remove all large or temporary files left by your job (here TEST001), to help conserve system resouces. Of course, if your job terminates abnormally this final clean-up step may never execute, leaving many files behind anyway. So to avoid filling the heavily used /var/tmp file system *on LC IBM machines*, run jobs on those machines in a special subdirectory that the batch system empties whenever and however your job ends (see the "Script Differences" section (page 15) below for details).

Between-Machine Script Differences

Some useful batch script features are limited by platform or vendor. The list below describes these useful but nonportable features, as supplements to the portable features in the annotated script in the previous section (page 10). Platform or vendor limitations are noted for each. (For additional PSUB options, some misleading and some compensatory, see the "Submit Your Job" (page 28) section below.) See also the warnings about using "large memory pages" on some LC AIX systems, and about "CPU affinity" performance issues on LC machines that have NUMA hardware, below (page 20). Also, not all environment variables are available on all platforms (see LC's Environment Variables (URL: http://www.llnl.gov/LCdocs/ev) manual for noteworthy local differences).

#PSUB -re	DEFAULT: The batch system automatically holds your error messages in a spool file until your job ends, where they could be lost. SUGGESTION: You must overtly request -re (write errors directly to your error file) if you want it, as shown here. WARNING: On LC machines, -re and -eo are considered conflicting PSUB options, so you must omit -re to use -eo (i.e., to consolidate error and log messages in one file). But if you chose not to combine your error and log files (-eo), then you should use -re to force error messages directly into your error file for safety.
#PSUB -c *constraint*	DEFAULT: The -c option specifies extra machine-level constraints on how your batch job runs. While -c itself is available on all LC machines, different constraints (-c arguments) are permitted on different machines. SUGGESTIONS: (1) Before using -c, consult both the "Plan Your Job Constraints" (page 20) and the "Submit Your Job" (page 28) sections below, so that you use only appropriate constraints and avoid unsupported ones on your target machine. (2) Do not confuse machine-level constraints (specified with -c) and node-pool constraints (specified only with -pool). For example, you can limit your job to LC's Thunder machine by using -c thunder, but you can limit it to the pdebug node pool only by (also) using -pool pdebug.
set timestamp	DEFAULT: The batch system does NOT automatically record the time at which each command in your job executes. SUGGESTION: You must overtly request time stamping if you want it, as shown here.
setenv *varname varvalue*	DEFAULT: On LC AIX machines, many Parallel Operating Environment (POE) environment variables are set by default or ignored by default to support MPI (between-process parallelization). See the "Local Defaults" section of the POE User Guide (URL: http://www.llnl.gov/LCdocs/poe) for a list. SUGGESTION: If you want to change the MPI defaults, or if your program uses POSIX threads (for within-process parallelization) and you need to set IBM/AIX threads-relevant environment variables (such as setenv AIXTHREAD_MNRATIO

4:1), include those commands in your batch script BEFORE you execute your main program, so that your preferred environment variable values will be available for program initialization. See the POE User Guide (URL: http://www.llnl.gov/LCdocs/poe) for possibilities.
WARNING: On BlueGene/L only, environment variables set in this standard way will *not* be visible to application programs launched using MPIRUN. To set environment variables for BG/L MPIRUN-executed programs, use the quoted, blank-delimited argument to MPIRUN's own -env option. For example:
mpirun -env "BGLMPI_ALLREDUCE=MPICH BGL_APP_L1_WRITE_THROUGH=1" ...

cd /var/tmp/$LOADL_STEP_ID

DEFAULT: The batch system usually does not clean up (destroy) any files left by your job when it ends, even if it ends abnormally. This sometimes completely fills /var/tmp, especially on the heavily used IBM machines.
STRATEGY: On *LC IBM AIX machines only,* LoadLeveler sets the environment variable LOADL_STEP_ID when it starts your batch job. The LC batch system (LCRM) "prolog script" then automatically creates a unique subdirectory for your job called /var/tmp/$LOADL_STEP_ID. If you CD into that special directory (as shown here) before executing the other steps in your batch job, then the batch system will automatically destroy all files left in that subdirectory when your job terminates, even if it terminates prematurely. This benefits all IBM users by greatly reducing the chance that /var/tmp on the IBM nodes will fill completely with orphaned job files. (Of course, you must promptly write to archival storage any job files that you really want to save, or they will also vanish with the special subdirectory if the job ends abnormally.)
WARNING: LC AIX machines that use SLURM instead of LoadLeveler do not support this feature.

grep NETWORK /etc/home.config

DEFAULT: Your job may need to behave differently in the OCF and SCF environments, but not be able to tell them apart.
SUGGESTION: One line in the system file /etc/home.config on every LC machine contains the keyword "NETWORK" (uppercase) and the (lowercase) value "ocf" or "scf" to always reveal your job's current network context. GREPing home.config, as shown here, returns the content of that line.

Running a Batch (LCRM) Job

Overview Chart

This chart summarizes the steps needed on LC machines to prepare, submit, run, monitor, and evaluate a batch (LCRM or Moab) job. Each section following the chart futher explains and illustrates one of the steps shown here.

```
                              ---------------
                              | Create your |
                              |   script    |
                              ---------------
                                     |
                        ---------------------------
                        |   Store input files,    |
                        | make output directories |
                        ---------------------------
                                     |
                              ---------------
                              |   Plan job  |
                              | constraints |
                              | (PLIM, etc) |
                              ---------------
                                     |
                              ---------------
                              | Submit your |
                              | job (PSUB)  |
                              ---------------
                                     |
        ---------------------------------------------------------
        |                            |                          |
------------------           -----------------          ---------------
| What is job's  |           |  How can you  |          | How can you |
|  status, id?   |           | alter the job |          | delete your |
|    (PSTAT)     |           |   features?   |          |    job?     |
|                |           |   (PALTER)    |          |    (PRM)    |
------------------           -----------------          ---------------
        |                            |                          |
        ---------------------------------------------------------
                                     |
                               --------------
                               | Check your |
                               |  log file  |
                               --------------
                                     |
                               ---------------
                               | Check your  |
                               | output file |
                               ---------------
```

Step 1: Create Your Script

Every use of the LCRM batch system begins with your running a text editor to make a shell script that specifies how your batch job will be handled (mostly by imbedded instructions to the PSUB utility) and exactly what programs to run (mostly using lines like you would type to run them yourself). For instance, the sample batch script shown here lays some groundwork, processes an input file with the UNIX SPELL program, and stores the output. This trivial project nevertheless typifies the features of most batch jobs, and you can see a detailed explanation for the role of each line here by consulting the Annotated Batch Script (page 10) section above. And Between-Machine Differences (page 15) warns about some known script portability pitfalls.

```
#!/bin/csh
#PSUB -eo
#PSUB -o /g/g16/jfk/mylogs/mylog1
#PSUB -nr
#PSUB -ro
#PSUB -tM 40
#job commands start here
set echo
echo LCRM job id = $PSUB_JOBID
mkdir -p /nfs/tmp1/jfk/myruns
cd /nfs/tmp1/jfk/myruns
ftp storage <<EOF
jfk
cd testdir
get test001
quit
EOF
/usr/bin/spell test001 >! sp.out
ftp storage <<EOF
jfk
cd outdir
put sp.out
quit
EOF
rm test001
```

After you design your batch script but BEFORE you submit it you need to prepare for your batch job in other ways (next section).

WARNING: the name of your batch-job script cannot contain a percent (%) character. PSUB rejects at submittal any job script whose name includes a percent character.

Step 2: Store Files, Make Directories

STORE YOUR INPUT FILES.
Most batch jobs, just like the little sample in the previous section, require data files as input while they run. Store such files in LC's archival storage system (storage.llnl.gov, called "storage" for short) before you submit your batch job. Because these input files are often large (and sometimes numerous), leaving them on disk makes them vulnerable to file purges or disk crashes that could prevent your job from getting them when needed.

Then, when your job runs, have your script retrieve your data files from storage as an early step. You can use passwordless FTP as an interface to the storage system, as shown in the foregoing script, or, for more reliability, you can use NFT (on LC production machines). For a comparative overview of basic, relevant FTP and NFT commands, see the EZOUTPUT File-Transfer Guide (URL: http://www.llnl.gov/LCdocs/ezoutput). For technical details on the storage-support features of NFT, consult the NFT Reference Manual (URL: http://www.llnl.gov/LCdocs/nft).

MAKE NEEDED DIRECTORIES.
Before you submit your batch script to LCRM you must plan to provide every directory and subdirectory needed to support your output file(s).

(1) Your error/log file(s) can go into a child directory of your common home (/g) directory, which is automatically shared among all nodes of all LC production machines. You should make this directory (e.g., /g/g16/jfk/mylogs) before you even submit your job. (A common mistake when using PSUB -ro (page 10) is to create error/log directories on only one node in a cluster, but in a file system such as /var/tmp that is local to each node. If you then allow the job to run on several (or all) nodes in the cluster and if LCRM picks one for execution that lacks the needed directories, your job dies almost immediately (as soon as -ro is encountered). Using a child of your common home directory for error/log output avoids this problem.)

(2) Running each batch job in a separate work subdirectory is good practice, free from the constraints on your home directory and sheltered from your other computing activities. To do this, create (MKDIR) the job subdirectory and move (CD) into it as the very first steps within your job's script. For example:

```
mkdir -p /nfs/tmp1/jfk/myruns
cd /nfs/tmp1/jfk/myruns
```

Note that on LC clusters with diskless compute nodes (such as the Atlas family of Linux/CHAOS machines), /var/tmp uses RAM on each node and to conserve memory CHAOS purges /var/tmp between all jobs. So using GPFS or Lustre, /nfs/tmp*, or archival storage is a much better choice for output. See the Annotated Batch Script (page 10) section above for more details.

Step 3: Plan Your Job Constraints

Before you submit your batch job, take account of the by-machine or by-bank resource constraints as well as the batch-system scheduling limitations that it will face and plan your submittal parameters to avoid trouble with those constraints. You can run the utility programs described in this section to get up-to-date reports on many of your job's diverse current constraints.

MACHINE IDIOSYNCRACIES.
(1) For example, on LC's UM, UV, UP, and Purple machines (all AIX clusters), most node memory is bundled in "large memory pages" but you must overtly enable your executables running there to use those large memory pages at the start of your job or receive a warning message for every task (see the "Enabling Large-Memory Pages" section (URL: http://www.llnl.gov/LCdocs/poe/index.jsp?show=s6.5) of the POE User Guide for details).
(2) On LC Linux/CHAOS clusters that have CPUs with "nonuniform memory access" (called NUMA hardware), application performance may vary greatly depending on whether or not processes are bound to the CPU where they start to run (CPU afffinity) and whether or not memory allocation is local to each NUMA node. See the "For Users of NUMA Nodes" section (URL: http://www.llnl.gov/LCdocs/chaos/index.jsp?show=s5.2) of the CHAOS reference manual for background on this issue and tools to help you manage it.

LCRM DOMAIN SCOPE.
LCRM divides SCF (secure-network) machines into two "domains." Users running PSUB on a machine in one domain *cannot* submit jobs to or monitor jobs on machines in the other domain (no domain crossover). See Step 4 (page 28) below for details.

ENVIRONMENT VARIABLES.
LCRM automatically passes to your job *some* environment-variable values that it captured on the machine where you submit the job, and it automatically sets *some* other environment variables in the job's execution environment. You may want or need to pass additional values to the job by invoking PSUB's -x option (page 31). For a thorough, comparative discussion see the long "Batch-Job Environment Variables" section (URL: http://www.llnl.gov/LCdocs/ev/index.jsp?show=s3.4) of LC's Environment Variables user manual. On BlueGene/L, you must use MPIRUN's own -env option to pass environment variable values to any application executed with MPIRUN. On LC machines where Moab has replaced LCRM for batch-job scheduling, environment variable handling depends on whether you submit your job using the PSUB emulator or the native MSUB tool. See the "Environment Variables" section (URL: http://www.llnl.gov/LCdocs/moab/index.jsp?show=2.3) of the Moab at LC user guide for details.

PLIM, LRMMGR, BRLIM, and SINFO

PLIM.
The PLIM utility, run with no options on machines using LCRM, reports the current value of nine seldom-changed system default limits that your job faces, including

- the maximum run (wall-clock) time for batch jobs,
- the maximum allowed job size (Mb),
- the default time limit your job gets if you do not specify one,

- the maximum allowable nodes and node hours for running a parallel batch job.

All times are unlabeled, but all have the form *hh:mm* (not *mm:ss*, so 0:30 is 30 minutes).

WARNING: if you run PLIM on a machine in a cluster (such as an open or secure Linux cluster), and if the job limits vary among nodes in the cluster, then PLIM's default report may not reflect the actual limits on nodes relevant to your job. To get an accurate limit list for a specific clustered node (*host*), use PLIM's (undocumented) -m option, as shown here:

```
    plim -m host
```

(for example, plim -m yana25). On LC machines using Moab instead of LCRM to schedule batch jobs, PLIM has been replaced by

```
    news job.lim.host
```

to report similar limit information.

LRMMGR.
An even longer labeled list of (seldom-changed) batch-system configuration limits that could affect your job (such as the maximum number of active jobs allowed per user or per local machine) is available by running LRMMGR (formerly called PCSMGR). LRMMGR (available on LCRM-scheduled machines only) reports almost three dozen limits, not all of which overlap with those PLIM reports and not all of which are highly relevant to typical jobs. But since hitting an unknown limit is a common reason for batch-job failure, this long limit list may help you troubleshoot problems.

Using LRMMGR calls for some patience. Start by typing LRMMGR and then respond to its interactive prompt (lrmmgr>) with one of these commands (note that if you pause for more than about a minute, LRMMGR warns you about inactivity and terminates itself if no input follows within 30 seconds):

help show	lists all available suboptions for the SHOW command, but without any explanation of their role.
show config *	lists the (several dozen) names for every job-control configuration set that LRMMGR is prepared to report. You can use any of these as suboptions for the SHOW command (next).
show *hostname*_config	reports a table of the current settings on the machine *hostname* for 25 LCRM configurable features, such as cpu time limit, maximum number of running jobs/user, maximum process size, and maximum and minimum nodes/job. System administrators can change these limits, but after February, 2006, such changes never affect already *running* jobs.

show thunder_config

is an example of using a specific configuration name to get a table of current settings for the specific cluster named (here, THUNDER).

show default_config

shows the current default settings for the machine where you are running LRMMGR (which may not be the machine where you intend to run your job). LCRM managers set these defaults by using LRMMGR to specify the attributes of a pseudo-configuration named TEMPLATE (all lowercase).

show feature *

lists the "features" (such as pdebug or pbatch node partitions) that you can use in the PSUB -c command to impose constraints on where and when your job will run.

show host *hostname*

lists the properties of the machine *hostname*, some of which are relevant to the way it runs batch jobs (included are the machine name, IP address, total number of CPUs).

For your convenience on all LC production machines that use LCRM, a text file that summarizes (some of) the LRMMGR output is available (in a nonstandardized format) at

```
/usr/local/doc/job.limits
```

Or you can see a tabulated, comparative version of (some of) the LRMMGR output for all OCF production machines at this (OTP-controlled) web site:

```
https://lc.llnl.gov/computing/status/limits.html
```

On LC machines where Moab has replaced LCRM to schedule batch jobs, LRMMGR no longer runs. You can get similar information by executing

```
mdiag -t
```

BRLIM.
Besides the local limits reported with PLIM and LRMMGR, you might encounter *global* limits (on jobs/user or jobs/bank) that span entire batch partitions. To see if any partition-wide limits exist where you plan to run your jobs, use the BRLIM utility (on LCRM-scheduled machines only), whose (tricky) options are explained in the Bank and Allocation Manual (URL: http://www.llnl.gov/LCdocs/banks). BRLIM does not run on nor report on machines whose jobs are scheduled by Moab, and Moab does not enforce BRLIM's global limits on resources.

SINFO.
On LC's Linux (CHAOS) clusters, open and secure, a SLURM utility called SINFO helpfully reports on the current status of the local compute nodes (availability, time limits, allocation state). SINFO's default report covers only general properties and broad node partitions. But you can invoke many customizing options (URL: http://www.llnl.gov/LCdocs/slurm/index.jsp?show=s4.4), as explained in the SLURM Reference Manual, to report on specific properties (such as disk space) of specific (sets of) nodes if this will help plan your job submittal.

PHSTAT (Production Host Status)

The tools described in the previous section (page 20) all reveal relatively stable, *seldom-changed* local limits (PLIM and LRMMGR), global partition limits (BRLIM), or cluster attributes (SINFO) that can constrain your job. But sometimes you need to know if current values of *dynamic LCRM attributes* (such as the current scheduler choice or scheduling cycle), changeable internal features of the "batch system" itself, are affecting your (planned or submitted) batch job.

In that last case, PHSTAT ("production host status") is the LCRM utility to try. PHSTAT runs wherever PSUB (page 28) runs (as a native tool, not Moab-emulated). If executed without options, PHSTAT reports a text table of LCRM-managed hosts (one for OCF, a different table for SCF) that reveals for each listed host:

- How LCRM now schedules that host (cluster backfill, memory backfill, multi-node backfill, or no backfill),
- The current status of several LCRM scheduling daemons,
- Whether LRMMGR's NORUNNEW feature has been turned on to block the start of jobs on that host,
- Whether timedout, staging, or terminating jobs are being scheduled,
- Total usable memory and percentage of memory already in use ("memory load"),
- The count of already committed nodes and total available nodes (but *not* the available node names), and
- Time (in seconds) since the local LCRM scheduler last ran.

If executed with the -t option, PHSTAT replaces its default memory and node report (above) with a table showing the current type and version of operating system and native batch system available on each LCRM-managed host. PHSTAT ends after displaying its report, from which you can optionally suppress the column headings (with -H) for easier postprocessing.

On LC machines where Moab has replaced LCRM for batch-job scheduling, most PHSTAT details are no longer relevant and PHSTAT does not run. You can get a little dynamic information from Moab by excuting

```
    mshow -a
```

BAC

The BAC utility reports the name of your (computer time) bank and three batch-relevant bank privileges:

ACCESS	tells whether you can merely use (U) the bank, or also expedite (E) jobs funded from it. See the Constraint Summary (page 27) below.
SC	reports your remaining number of authorized "short production" days, usually 0. See the Constraint Summary (page 27) below.
EC	reports your remaining number of authorized "job expediting" days, usually 0. See the Constraint Summary (page 27) below.

BT and RDBSEE (Replaced)

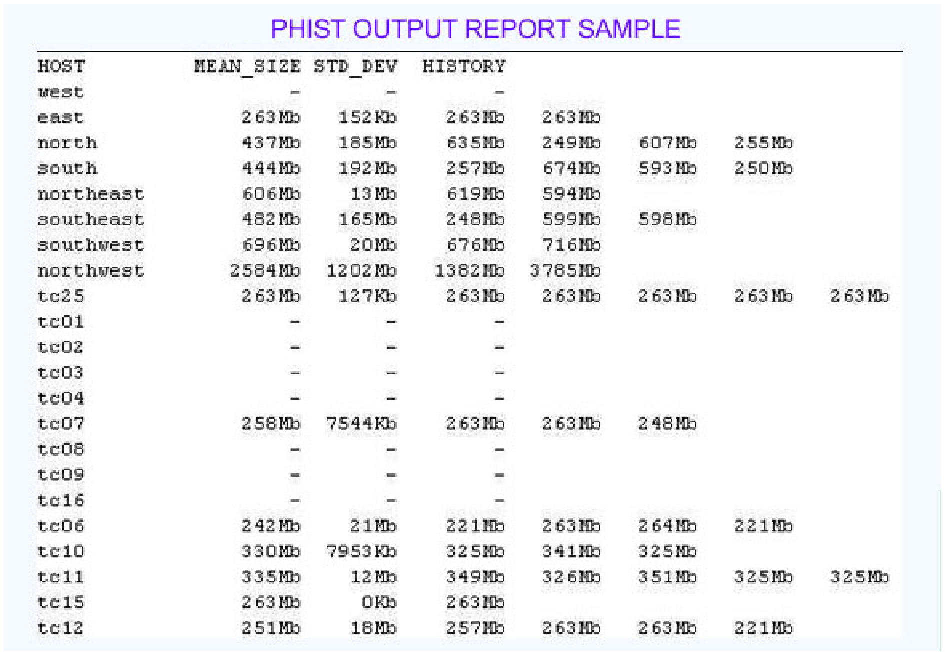
The former BT utility reported the computer time (in the form *hh:mm*) for your bank that was ALLOCATED, USED, and still AVAILABLE. With the start of fair-share scheduling in 1998, the PSHARE (page 26) utility now fills somewhat the same role that BT once did.

The former RDBSEE utility with the -B option reported data from the Resource Data Base, including your default (DEFPRIO) and maximum (MAXPRIO) "local priority" (both 0 for most users). Ironically, DEFPRIO, MAXPRIO, and the "local priority" they control (and that some users can specify with PALTER's -p option) still pertain to LC production machines where fair-share scheduling is used. But you can no longer run RDBSEE to discover your DEFPRIO or MAXPRIO values, and your fair-share priority is completely independent of and different from that "local priority." See the PSHARE (page 26) section below for some suggestions.

PHIST

PHIST helps you plan future jobs by summarizing the actual amount of memory used by recent past jobs (yours or someone else's). The PHIST utility lists job-memory-size statistics and history. This information includes the average or mean job size and standard deviation of job sizes as well as up to the five most recent job sizes listed in order of termination. The data sizes are based on the maximum resident set size attained during the jobs' execution. On multinode hosts, the maximum resident set size for a job is defined as the maximum resident set size attained on any of the nodes on which the job executes.

A sample PHIST report is shown in Figure 1.



PHIST OUTPUT REPORT SAMPLE

```
HOST          MEAN_SIZE STD_DEV  HISTORY
west                -       -        -
east            263Mb   152Kb    263Mb    263Mb
north           437Mb   185Mb    635Mb    249Mb    607Mb    255Mb
south           444Mb   192Mb    257Mb    674Mb    593Mb    250Mb
northeast       606Mb    13Mb    619Mb    594Mb
southeast       482Mb   165Mb    248Mb    599Mb    598Mb
southwest       696Mb    20Mb    676Mb    716Mb
northwest      2584Mb  1202Mb   1382Mb   3785Mb
tc25            263Mb   127Kb    263Mb    263Mb    263Mb    263Mb    263Mb
tc01                -       -        -
tc02                -       -        -
tc03                -       -        -
tc04                -       -        -
tc07            258Mb  7544Kb    263Mb    263Mb    248Mb
tc08                -       -        -
tc09                -       -        -
tc16                -       -        -
tc06            242Mb    21Mb    221Mb    263Mb    264Mb    221Mb
tc10            330Mb  7953Kb    325Mb    341Mb    325Mb
tc11            335Mb    12Mb    349Mb    326Mb    351Mb    325Mb    325Mb
tc15            263Mb     0Kb    263Mb
tc12            251Mb    18Mb    257Mb    263Mb    263Mb    221Mb
```

Here, the first column shows the host name, for example, west and east. The second column shows the size of the job run, and the third column shows the difference in the size of jobs run. The last column shows the sizes of the last jobs run up to a maximum of five jobs.

To use PHIST (only on machines scheduled by LCRM, not Moab), type

```
phist
```

at /usr/local/bin/.

PHIST's most useful control options are:

-c *constraint*	shows only jobs that meet your specified constraint. If you are not familiar with constraints, refer to Step 4: Submit Your Job (page 31).
-u *username*	compares your jobs with those of other users or shows their memory histories even if you have no recent jobs of your own.

PSHARE

On all LC production machines, "fair-share" political scheduling (page 7) has replaced time-allocation political scheduling. This significantly affects your constraint planning for batch jobs because under the fair-share approach:

- Shares (your entitlement to compute resources) are NOT decremented or exhausted as you do work, in contrast to a fixed allocation of CPU minutes tied to a shift. Instead, your shares and your past usage (of CPU time) together determine your (constantly changing) scheduling priority.
- Just as your current fair-share priority is computed from your AGGREGATE shares and usage, it applies to ALL your jobs. It varies with time (see next item), but at any one time all your jobs funded from the same bank for the same set of machines (LCRM partition) have the same fair-share priority, regardless of job features.
- Both your shares and your usage are normalized (divided by sums) over the set of currently ACTIVE users. This means that while traditional job priorities were independent of who was logged in, fair-share priorities vary greatly and sometimes suddenly as the instantaneous user set changes on a machine. LCRM recomputes your fair-share priority once every minute and checks your usage every 540 seconds.
- The "local priority" that some users are allowed to set using PSUB's -p option affects only how your job competes with other jobs in the same bank. Despite the name, this is completely independent of and different from your (more important) fair-share priority.

The PSHARE utility (on LCRM-scheduled machines only) fills roughly the same role for fair-share scheduling that the BT utility filled for traditional time-allocation scheduling: it gives you indirect clues about how your job(s) will compete against the work of others. PSHARE reports your (raw and normalized) shares, your nomalized aggregate usage, and (optionally) your current fair-share priority, and you can request these values for other users (with -u) or banks (with -b) as well. For a description and analysis of the assumptions and algorithms that PSHARE relies on for its normalization, usage decay, and priority calculations, see the Fair Share Scheduling (URL: http://www.llnl.gov/LCdocs/dpcs/index.jsp?show=s5) section of the LCRM (DPCS) Reference Manual.

Among PSHARE's most useful options (usually used simultaneously) are:

-p	reports each user's current fair-share priority, a decimal number between 0 and 1 inclusive that reflects relative likelihood to get more compute resources.
-t *bankname*	reports on the specified bank (usually your bank) and all its authorized users. This is the set of people whose (current but often-changing) active/nonactive status most affects your fair-share priority.
-0	(zero, same as -O, uppercase oh) omits from PSHARE's report users with no current usage. This prunes the (usually many) nonactive users, whose normalized shares and priority are always 0 by definition, yielding a much easier to interpret report.

Alternatively, you may want a report on the current status of all the parents of your bank, since your bank hierarchy (page 8) strongly affects the "normalized value" of your shares. In that case, use PSHARE with the option

-r *bankname*	reports the shares, normalized shares, and (aggregate) normalized usage for the specified bank (usually your own bank) and all of its parents up to the top of the bank hierarchy.

To avoid confusion, PSHARE lets you report exclusively on either of two sets of parameters:

-sched	(the default) reports the normalized shares, priorities, etc., only for batch jobs scheduled by LCRM.
-active	reports the normalized shares, priorities, etc., used as session parameters by all active sessions, both interactive and batch.

On LC production machines where Moab has replaced LCRM as the across-cluster batch-job scheduler

```
mdiag -f
```

has replaced PSHARE as the tool for reporting detailed share-like usage information.

Constraint Summary

Once upon a time, your batch job's "local priority" (assigned with PSUB's now-defunct -p option) affected how your job competed locally, with other jobs funded from the same bank. Your batch job's short-production status (assigned with PSUB's former -sp option) or its "expedite" status (assigned with the PEXP utility) affected how your job competed globally, with jobs funded from other banks. Today, however, LCRM changes have disabled or greatly altered all of these once-important scheduling factors for typical batch jobs on most LC production machines.

In practice, for most users, factors unrelated to the specific features of your individual job determine more than anything else how your job competes against other jobs waiting to run in the batch system:
(1) The bank you draw on and its hierarchical relationships (page 8) to other banks strongly affect how your shares and your past usage are "normalized" in fair-share scheduling calculations.
(2) Changes in the mix of currently active users (URL: http://www.llnl.gov/LCdocs/dpcs/index.jsp?show=5.1.2) can cause large, sudden changes in your influential fair-share priority, over which you have little control.
(3) Local limits (page 20) on the number of simultaneous jobs per machine or per user can sometimes significantly affect when or where your job will run. Global limits, applied to a whole batch partition, can interact intricately with local limits too. See the beginning of Step3 (page 20) for how to discover these limits.

For a more detailed look at the assumptions underlying LC scheduling policy, see the Job-Scheduling Policy (page 7) section above. For details on the fair-share priority algorithms and their hidden implications for your work, see the Fair Share Scheduling (URL: http://www.llnl.gov/LCdocs/dpcs/index.jsp?show=s5) section of the LCRM (DPCS) Reference Manual.

Step 4: Submit Your Job

On LLNL Machines.

PSUB:
Run PSUB to submit your job, taking note of the crucial (page 30), the helpful (page 31), and the sometimes misleading (page 39) PSUB options explained here (and illustrated at the end of this section). A separate subsection (page 34) compares the special PSUB options that you need to manage the detailed execution of a parallel job.

On LCRM-scheduled machines, PSUB is a native tool whose options are explained in this section. On Moab-scheduled machines, however, PSUB is only an emulator and only a subset of its options are automatically translated into MSUB options. To see which PSUB options MSUB supports, and for information on an LCRM-to-Moab script converter as an alternative, consult the "PSUB Options Conversion" section (URL: http://www.llnl.gov/LCdocs/moab/index.jsp?show=s2.5) of the Moab at LC user guide.

OPTION PRIORITY:
PSUB options that you want to invoke with every run, such as output control options, can be imbedded at the start of your job script file, as explained in the Annotated Batch Script (page 10) section above. If you include the same PSUB option both on your execute line and in your script file (with different arguments), the version on the execute line takes precedence over the version in your script.

EXPEDITING:
Only LCRM managers or others specifically authorized by them can use special PSUB options to expedite a job, exempt it from the usual job limits, or force its priority to a preassigned value. For instructions on these privileged uses of PSUB, see "Expediting and Exempting Jobs" in the LCRM (DPCS) Reference Manual (URL: http://www.llnl.gov/LCdocs/dpcs/index.jsp?show=s4.11).

SCRIPT VERSIONS:
When LCRM accepts your submitted job (by returning a job id number) it copies your script (and other metafile information) from the submitting machine to LCRM's own "control host." This avoids start-up delays, but it means that changes you make to *your* copy after this point will NOT affect how the (original "spooled") script executes. So the time to correct flaws in your script is *before* you submit it to LCRM, not afterward.

SCF DOMAINS:
All OCF (open-network) machines scheduled by LCRM lie in a single scheduling domain: you can submit (PSUB) a job on any OCF machine to run on any other, or monitor (PSTAT) any job from any machine. Starting in September, 2005, however, LCRM divided SCF (secure-network) machines into *two* disjoint scheduling domains (based not on operating system but on underlying batch system):

- All LoadLeveler machines (UM, UV, Tempest, etc.; these are also all AIX machines), and
- All SLURM machines (ACE, QUEEN, LILAC, PU, etc.). Note that PU is an AIX machine that nevertheless uses SLURM rather than LoadLeveler to manage its jobs.

The practical impact of the SCF domain split is that LCRM tools (such as PSUB and PSTAT) only work with hosts within a single domain, not across domains. Consequently, on SCF, you *cannot* submit or monitor LoadLeveler jobs from SLURM machines or vice versa.

SRUN ROLE:

LLNL Linux (CHAOS) machines even *without* LCRM/DPCS still provide a way to submit parallel jobs to a *local* queue managed by LC's Simple Linux Utility for Resource Management (SLURM). SLURM's SRUN tool (URL: http://www.llnl.gov/LCdocs/slurm/index.jsp?show=s4.2) enables elaborate (but purely local) control of your job's nodes, task distribution, I/O handling, and other resource use. See the SLURM Reference Manual (URL: http://www.llnl.gov/LCdocs/slurm) for examples of using SRUN to launch such "local batch" jobs.

Elsewhere.

Offsite, nonLLNL ASCI collaborators can submit their collaborative MPI projects to run on the ASC Tri-lab machines (when available) by using GLOBUS (rather than PSUB) as a remote front end to LLNL's local batch system. The GLOBUS project is a team effort to promote distributed supercomputing by Argonne National Laboratory, NCSA, NASA, the University of Chicago, and the University of Southern California. To prepare for future GLOBUS-based use of ASC Tri-lab machines, you can review general project materials at the www.globus.org (URL: http://www.globus.org) web site, or download a PDF or PS version of the Globus Quick Start Guide (36 pages) by following the instructions at

```
http://www.globus.org/documentation/quick_start.html
```

Significant inside information and local modifications are needed, however, to actually invoke GLOBUS to submit jobs to LC. Consult the Globus User Guide (for LC) (URL: http://www.llnl.gov/LCdocs/globus) for a localized list of step-by-step instructions, relevant hints about using GLOBUSRUN in place of PSUB, and a summary of the most useful features of the GLOBUS Resource Specification Language.

Crucial PSUB Options

-b *bankname*	specifies the bank to draw CPU time from when your job runs (you must be authorized to draw against the bank you specify). Everyone has a default bank now, but it may not be appropriate for charging your batch jobs on all machines. So overtly specifying your desired bank when you submit each job is good practice.
-lM *size*	specifies your job's memory needs, but only in an optional, advisory way. See the "Misleading Options (page 39)" section below for a full analysis of -lM's real role and syntax.
-lt *mmm*	specifies the maximum CPU time (in minutes) for each process in your batch job. A process that exceeds this time limit is killed. (To specify hours instead of minutes, use the form *hh:mm* or use an h suffix, as in 5:00 or 5h.) PSTAT reports this value as "time limit per CPU." To specify a total-job (or, on most LC machines, per-CPU) instead of a per-process time limit, use -tM. WARNING: once you specify a limit with PSUB's -lt, you *cannot* later change it by running PALTER (see -tM below as an alternative that you can change with PALTER).
-tM *mmm*	specifies the maximum CPU time (in minutes) for your entire batch job (on SMP systems, such as ACE or QUEEN) or per-CPU (on "multi-node" systems, including virtually all other LC production machines). A job that exceeds this time limit is killed or checkpointed. (Remember that -tM 1000 means 1000 minutes. To specify hours instead of minutes, use the form *hh:mm* or use an h suffix, as in 5:00 or 5h.) A 4-CPU job slated to run for 1 hour would need -tM 4h on an SMP system, but it would need -tM 1h on an LC multi-node system. Starting in February, 2006, the -tM time limit *includes* idle time on allocated but not-used nodes, and it is reported by PSTAT as "time charged." You can use both -lt and -tM if you want to impose both kinds of limits on the same job. And, if you start a job and forget -tM, you can change your job's default total time limit (usually 30 minutes) to the value you would have specified with -tM by running PALTER (page 45) after you submit the job (but you cannot use PALTER to change a limit set with -lt). [You can optionally limit your job's wall-clock time as well as its CPU time; see -tW in the next section.]

Helpful PSUB Options

-c *constraint* limits the set of *hosts* (machines) on which your job can run to only those that have the feature(s) that you specify with *constraint*. Even if a requested feature happens to be the name of a node pool (e.g., pbatch), however, -c *never* also limits your job to that set of nodes (you must instead use the -pool option along with -c). So -c only selects machines and -pool only selects node sets. See -pool below. By default, a job runs on the machine where it was submitted.

You can conjoin two constraints by using the form
-c "con1&con2"
or disjoin two constraints by using the form
-c "con1|con2"
Note that no extra space is allowed within the -c argument, so that "yana | 2048Mb" is INcorrect syntax.

MULTIPLE CONSTRAINTS:
PSUB accepts only one -c option per job. If a script contains several -c option lines, all are discarded (without warning) except the LAST one. If one -c option appears on PSUB's execute line, that will override the last -c option within the submitted script. The ONLY way to impose multiple constraints on the same job is to use one -c option and conjoin (&) or disjoin (|) its multiple arguments within quotes, as shown in the syntax above.

DISCOVERING CONSTRAINTS:
The name of each production machine is a constraint and -c is most often used to limit job execution to one or more machine(s) you specify. For example, you could use -c "yana25|yana26" on the open YANA cluster. If a machine supports any other constraints you can discover them by running the LRMMGR utility and responding to its prompt with the string

```
show host machinename
```

At the end of LRMMGR's multiline report is a field called "assigned features" that lists the constraints you can use with -c on that machine (only).

DEFAULT CONSTRAINTS:
LCRM managers can use the DEFCONST option of LRMMGR to specify default constraints for each partition (group of machines managed by LCRM). LCRM then treats every job submitted with no -c PSUB option as if it had been submitted with the default constraints.

THE Mb CONSTRAINT:
You can specify your job's per-node memory needs by using the attribute-laden version of PSUB's -ln option. (page 34) Formerly, you could specify your job's total memory needs by using a number followed by the string "Mb" as a constraint (an argument for -c). Now, however, Mb represents a purely undefined feature (*not* memory) that some LC machines happen to accept as a constraint. For example, you could specify

```
   psub -c "lustre&2048Mb"
```

which would limit your job to run only on machines that have the Lustre file system and at least 2048 units of something called Mb, but this has no effect on memory requirements for your job.

SRUN CONSTRAINTS:
On LC Linux (CHAOS) machines, the local job-management tool SRUN (URL: http://www.llnl.gov/LCdocs/slurm/index.jsp?show=s4.2.7) supports eight additional constraint options (for example, a genuine memory constraint plus others on which nodes to use and avoid, and on node contiguity) that let you impose much more fine-grained (but purely local) constraints on your parallel Linux jobs. You can also use SRUN within LCRM scripts.

-np *cpn*
(default is 1, replaces -cpn) specified CPUs per node on cluster-scheduled machines (such as GPS, no longer in service at LC). On node-scheduled machines (IBM/AIX or Linux/CHAOS clusters) -np is just advisory and is always dominated by any CPUs-per-node information contained within the argument(s) of the -ln option (see the next subsection (page 34) for details).

-pool *pname*
(default varies by machine) limits the set of *nodes* (within a multinode "host" or cluster, but never the set of hosts) on which your job can run to the pool or partition called *pname* (e.g., pviews). LCRM first uses -c (see above) to limit your job to specified *hosts* (machines), then separately uses -pool to limit it to a specified node pool (partition) on any hosts that qualify. Even if a node pool name is also a possible machine constraint (e.g., pbatch), -pool alone never selects *machines* on which to run your job. Without -pool, LCRM uses the target machine's default node pool.

-prj *projectname*
designates *projectname* as an optional, convenient label (up to 127 characters) for your job (reported in PSTAT -f reports and takes the place for tracking purposes of the former "account" attribute that was eliminated in August, 1005).

-s *shellpath*
(lowercase ess) specifies the absolute pathname of the shell (e.g., /bin/csh) that your job should use. Including the string #!*shellpath* as your script's first executable line is a more reliable way to do this. Note that MSUB uses -S (uppercase ess) to specify the job's shell.

-standby	places your job in STANDBY (S) class, that is, with such a low scheduling priority that it runs only when no normal or expedited jobs are available to run. LCRM terminates standby jobs as soon as any higher priority job arrives (with grace time only if the job registers for a signal; see the LCRM (DPCS) Reference Manual's (URL: http://www.llnl.gov/LCdocs/dpcs) "Class Values" section for standby termination details and workarounds). You can change the standby status of a *queued* job (but not one that has started running) with PALTER. (page 45) (Not all machines allow STANDBY jobs; see "Discovering Constraints" above for how to discover if STANDBY is allowed on a specific machine.)

-tW *mmm*

specifies the maximum wall-clock time (officially called "elapsed run time") in minutes for your entire batch job. A job that exceeds this time limit is killed or checkpointed. LCRM defines elapsed run time as the difference between the time when your job completes and when it starts to execute. Using -tW is always optional, and is logically independent of -tM (maximum CPU time), but the default value of maximum elapsed run time depends on your -tM setting on some machines, as this chart shows:

```
Machine   Elapsed run time -tW limit
          Default        Maximum
------------------------------------
IBM SP    Equals time    Equals maximum
          specified      CPU time limit
          with -tM       for job pool
          (30 min if
          -tM omitted)

Linux     Unlimited      Unlimited
------------------------------------
```

PSTAT (with -f) reports your -tW setting (if any) as "elapsed run time limit." Use the format *hh:mm* to specify hours instead of minutes, as with -tM.

-v

(verbose) displays nonfatal PSUB warning messages (by default PSUB reports only errors that prevent submittal of the job).

-x

(lowercase eks) passes specific environment-variable values (that you specified in your dot files or interactively) from the machine where you *submit* your batch job to the machine where it *executes* (they are not all passed by default). PSUB-*string* environment variables capture some aspects of your submittal environment, and LCRM sets about two dozen other environment variables by default. But all of your additional environment-variable customizations are passed to your job only if you invoke -x. ENVIRONMENT is always set to (the uppercase string) BATCH on the execution machine, and the value of LD_LIBRARY_PATH is never passed even if you invoke -x (instead, set it within your job script if you need a nondefault value). For more details on how -x works, see the "How LCRM Uses Environment Variables" section (URL: http://www.llnl.gov/LCdocs/ev/index.jsp?show=s3.4.2) of LC's Environment Variables user guide. MSUB users can get the same result by invoking MSUB's -V (uppercase vee) option, while -v (lowercase vee) passes only specified environment variables.

Parallel PSUB Options

WARNING: if your batch job runs a massively parallel program but directs all output to your common home directory (or to any globally mounted file system), this flood of network traffic to the NFS-mounted file server will probably degrade performance not only for you but for *all users on all machines* that share that file system. Responsible computing means sending your parallel-code output only to the dedicated *parallel file system* that is mounted on just your execution machine. See LC's I/O Guide (URL: http://www.llnl.gov/LCdocs/ioguide) for specific advice.

These PSUB options serve exclusively to manage in detail the way that a parallel job executes in batch on a massively parallel machine.

Generally Available:

-ln *arg*	(REQUIRED for all parallel jobs on multinode machines; without -ln the job runs as a single-node serial job.) The -ln option specifies how many nodes the job needs, and (optionally) how needed features (such as memory or CPUs) should be distributed among those nodes. PSUB accepts -ln in either of two formats (both explained below): the first is attribute free and the second is attribute laden (so the syntax for *arg* varies between them). See also -pool in the previous section (page 31).
-ln *num*	requests *num* nodes without specifying the features or attributes of those nodes. Here *num* may be: (1) a positive integer (example: -ln 8), which requests exactly that number of nodes, or (2) a hyphen-separated pair of integers (example: -ln 48-64), which requests at least the first number of nodes and at most the second. LCRM assigns to such range requests the largest number of nodes that are free (within the specified range) such that no higher priority job is delayed by the assignment.

-ln '*minnodes*[-*maxnodes*] (*attlist*)'

(NOTE the single quotes that begin and end the argument of -ln here; they protect enclosed characters from interpretation by your current shell) specifies a number of nodes and a list of quantified attributes or features distributed among those nodes, where:

minnodes	is the smallest number of nodes that this job needs (a positive integer).
maxnodes	is the largest number of nodes that this job needs (a positive integer greater than *minnodes*.) If you need a specific number rather than a range, use only *minnodes*.

attlist is a parenthesis-enclosed, comma-delimited list of attribute assignments to nodes, where each member of *attlist* has the format

nodect[:(*quan att, quan att,...*)]

For example, 2:(8cpn,4Gb), 4:(4cpn,16Gb), 24,...
Here the subcomponents of each *attlist* item are:

nodect is the number of nodes that need to have the atttibute(s) specified by the *quan att* pairs after the colon (a positive integer).

quan is the quantity for each attribute (a positive integer, such as the number of CPUs per node or the amount of memory). If the attribute is an abstract unitless feature (such as "gcard"), omit *quan* (such as 2:gcard).

att is the attribute assigned to (required of) the *nodect* nodes. LCRM now accepts as attributes either abstract unitless features (such as "gcard") or either of two predefined features, namely:

(1) CPUs PER NODE.
This is always expressed as a number (*quan*) followed by the string CPN (such as 8cpn). You cannot use both the CPN attribute here and the separate -cpn PSUB option on the same execute line.

(2) MEMORY.
This is always expressed as a number (*quan*) followed by a string that indicates the byte units. Acceptable byte-unit choices are:
GB|Gb|gb|MB|Mb|mb|KB|Kb|kb
(example: 16Gb). These units increase in 1000-fold steps except that 1Kb = 1024bytes. The previously used separate Mb PSUB *constraint* (-c) is still accepted but it no longer specifies memory size (merely an undefined feature called "Mb").

EXAMPLES:

```
        -ln '48(24, 24:16Gb)'
```

(note the enclosing single quotes) requests exactly 48 nodes of which 24 have no feature requests and 24 must have 16Gb of memory.

```
        -ln '8-48(2:(8cpn,4Gb), 3:(4cpn,8Gb), 1:gcard)'
```

(note the enclosing single quotes) requests at least 8 but no more than 48 nodes, where 2 nodes have the first set of attributes (8cpn and 4Gb memory), 3 have the second set of attributes, and 1 has the unitless gcard attribute.

GOALS:
(1) The *node-range* versions of -ln address the needs of those whose codes can scale at run time to use as many suitable nodes as happen to be available when LCRM schedules the job.
(2) The *attribute-laden* versions of -ln support jobs that run on heterogeneous clusters, where node features vary significantly and the job can only run successfully on a node subset with the specific combination of features that it needs.

IMPLEMENTATION:
(1) LoadLeveler--On IBM SP systems, LoadLeveler will support attribute-specified "heterogeneous runs" as soon as its API scheduler is operational (a future release).
(2) SLURM--On LC Linux systems, SLURM can assign features to nodes but it cannot yet quantify (count) those features when allocating nodes to jobs.

-cpn *num* (deprecated, better is -np) specifies the CPUs per node that this job requires on every node that it uses. If your job requires different numbers of CPUs on different nodes (a "heterogeneous job"), use the CPN attribute in the attribute-laden version of the -ln option (above) instead. You cannot use both -cpn and the -ln CPN attribute on the same PSUB execute line (and any latent conflicts are resolved in favor of -ln).

-g *[tasks] [switch] [@ layout]*
-g *cpucount*

(optional) specifies the job's "geometry," now used by LCRM (only) to calculate the job's tasks/node. Option -g is implemented for AIX machines, regardless of whether they use LoadLeveler or SLURM as the underlying job-control system, but it is not implemented for LC Linux/CHAOS machines. Here:

tasks	is the total number of tasks for this job (default is 1).
switch	(ignored on Compaq machines, used only on IBM SP machines) is the type of communications switch desired for this job, which may be either

ip	specifies Internet Protocol.
us	specifies User Space (the default).

layout specifies how the job's tasks are spread among its nodes, where the choices are:

tpn *num*	specifies *num* as the tasks per node, or
dist	spreads the tasks as evenly as possible among the nodes (the default).

cpucount (BG/L only) specifies the number of CPUs desired on each 1024-CPU BG/L node, where the only allowed values are 256 (a quarter node) or 1024 (all other values are "rounded up" to either 256 or 1024).

-net *protocol* (optional) specifies your job's between-threads communications protocol, where the choices are MPI (the default and usually best choice) or LAPI (a low-level API library for those few who want to develop their own alternative to MPI).

-nettype *adapter*

(optional) for IBM machines with two network adapters per node ("double-single" architecture, such as Frost, Ice, and White), specifies which network adapter to use, where the choices are csss, css1, and css0 (the default, and the only choice on one-adapter IBMs). NETTYPE is ignored on all nonIBM machines.

-nobulkxfer disables the use of Remote Direct Memory Access (RDMA). RDMA speeds MPI communications for some parallel codes running on IBM Power4 AIX machines (only applies to UM and UV at LC). RDMA, where it exists, is on by default.

Available on BlueGene/L ONLY:

-bgl "*srun-options*"

accepts as attributes control information that LCRM passes directly to SLURM's SRUN tool to manage your job in ways that only pertain to the special resources of BlueGene/L. The whole -bgl attribute list must be quoted and comma delimited with no internal spaces (e.g., "norotate,node_use=coprocessor"). Each of these (optional) -bgl attributes maps to one BG/L-only SRUN option with the same name:

geometry=*N[xM[xO]]*

specifies your job's size in "nodes" in each direction within BG/L's field of nodes (e.g., geometry=1x2x4 for 8 nodes). SLURM regards each BG/L 512-node dual-processor "base partition" as a *single 1024-processor node.* Use SLURM's SMAP utility on BG/L to visualize job layout and the geometric intermixing of several jobs.

If you omit this geometry attribute, then PSUB uses 1x1x1 as the default (or if you also use -ln *num* then PSUB uses *num*x1x1 as the default). If you omit O then the default geometry is NxMx1; if you omit both M and O then the default is Nx1x1.

conn_type=mesh|torus

specifies the type of interconnect that you want used between BG/L "base partitions" ("nodes" to SLURM), where the choices are mesh (the default) or torus.

node_use=coprocessor|virtual

specifies how to use the *second* processor on each BG/L compute node, where the choices are coprocessor (the default, so that the processor number is always t0) or virtual (allows processor numbers t0 and t1, but seems to be incompatible with the TotalView debugger).

[no]rotate	disables rotation of job geometry to fit available space (the default is to rotate).

Misleading PSUB Options

-lM *size*	(optional, advisory) estimates a job's per-node (high-water-mark) memory needs. The LCRM scheduling algorithm now uses this advisory guess of memory needed, along with historical data on the actual memory used by your five most recent batch jobs, to minimize the risk of memory oversubscription when it plans when and where to run your job. Jobs that exceed their specified -lM memory advisory are *not* terminated, however, so this is *not* a rigid maximum as are the -lt and -tM time limits discussed in the "Crucial Options (page 30)" section above. (Note that you can specify your specific per-node memory needs by using PSUB's -ln option. (page 34) Or you can run PLIM (page 20) to discover the maximum allowed size for a batch job on the machine where you plan to run.) For -lM, *size* always has two contiguous parts (*mmmuu*):

mmm	is the memory-needed guess in digits (e.g., 500).
uu	is the unit of memory, where the choices are b, kb, mb, or gb (for bytes, kilobytes, megabytes, or gigabytes). Thus 25gb specifies 25 gigabytes of memory.

-ln *num*	does NOT specify the NICE value of your job as it does in some other UNIX environments, but rather -ln at LC specifies the minimum number of nodes required for a parallel job, (page 34) and optionally the attributes that those nodes must have. NICE values are not an important feature of LC batch jobs.
-p *prio*	(reserved) PSHARE (page 26) reports your job's computed fair-share priority, which authorized users (only) can override by using PSUB's (or PALTER's) privileged -p option (see the "Forcing Job Priorities" section of the LCRM (DPCS) Reference Manual (URL: http://www.llnl.gov/LCdocs/dpcs) for details). If you are not authorized but try to use -p anyway, LCRM still accepts your submitted job but it automatically ignores your priority request.

PSUB Examples

When you submit a job with PSUB, remember that the maximum length of an LCRM job name (and hence a script name) is 15 characters (although user names can be 31 characters and -prj "project names" can be 127 characters).

(1) A typical use of PSUB to request a 10-CPU-hour run of the batch script PROJ23 (on any available node in the cluster from which it was submitted) might be:

```
psub -b sci -tM 10:00 proj23
```

(2) A typical use of PSUB on the YANA cluster to force PROJ24 to run on either YANA25 or YANA26 (nodes with high CPU time limits) for no more than 6000-minutes/process might be:

```
psub -b sci -lt 6000 -c "yana25|yana26" proj24
```

Note here that PSUB does NOT allow extra space within the -c constraint argument, so that trying "yana25 | yana26" would be INcorrect syntax.

(3) A typical PSUB execute line to submit a parallel job PROJ25 to run with 16 tasks (g) on 8 nodes (ln) of THUNDER (c) for 5 hours (tM) might be:

```
psub -b xyz -tM 5:00 -ln 8 -g 16 -c thunder proj25
```

(4) A typical PSUB execute line to submit a parallel job PROJ26 to run on 10 nodes (-ln) in the pviews partition (-pool) of the THUNDER cluster (-c) in standby (very low priority) mode might be:

```
psub -b xyz -ln 10 -c thunder -pool pviews -standby proj26
```

(5) Two equivalent PSUB execute lines to submit a parallel job PROJ27 on 48 nodes each of which must have at least 4 CPUs per node are:

```
psub -cpn 4 -ln 48 proj27

psub -ln '48 (48:4cpn)' proj27
```

Note the single quotes in the second version. To also specify at least 4 Gb of memory per node, use the form

```
psub -ln '48 (48:4cpn,4Gb)' proj27
```

(You can run PALTER (page 45) after you submit your job to adjust some, but not all, of the job features that you assigned originally with PSUB.)

Environment Variables and Job Submittal

The Livermore Computing Resource Management (LCRM) system uses many, diverse environment variables to manage your batch jobs and to run those jobs successfully on your target machine. LCRM divides the batch-relevant environment variables into four disjoint sets (and you can optionally set others and pass their values to your job when you submit it if you wish).

For every batch job that LCRM manages, it interacts with four disjoint sets of environment variables as follows (independent of any others declared by each job's owner):

```
A. SUBMITTAL environment variables--
   Those that LCRM creates from your job and its
   environment when you run PSUB.
B. EXECUTION environment variables--
   Those that LCRM sets for your job in the job's
   execution environment, automatically.
C. UNSET environment variables--
   Those that LCRM unsets for your job
   when the job executes.
D. DEPRECATED environment variables--
   Formerly important for LCRM but now replaced
   by others in sets A, B, or C.
```

Using PSUB's -x option (page 31) lets you pass to your job any additional, customized environment-variable values not covered in A, B, C, or D above but set by you before you run PSUB (on the submittal machine). For a detailed comparison of which environment variables fall into each group and the order in which LCRM sets those environment variables when your job begins to execute, see the long "Batch-Job Environment Variables" section (URL: http://www.llnl.gov/LCdocs/ev/index.jsp?show=s3.4) in LC's Environment Variables guide.

On LC machines where Moab has replaced LCRM for batch-job scheduling, environment variable handling depends on whether you submit your job using the PSUB emulator or the native MSUB tool. See the "Environment Variables" section (URL: http://www.llnl.gov/LCdocs/moab/index.jsp?show=2.3) of the Moab at LC user guide for details. PSUB users can still invoke -x to pass all environment variables from the submittal to the execution environment, while MSUB users can get the same result by invoking MSUB's -V (uppercase vee) option. MSUB's -v (lowercase vee) passes only those environment variables that you specify.

Step 5a: Monitor Your Job

You can discover your batch job's unique LCRM identifier (its JID), monitor its current status, and remind yourself of its attributes (some of which you can edit with PALTER (page 45)) by using the PSTAT utility. If run without options, PSTAT reports a line of column headers and then a one-line summary for each not-finished batch job submitted *by the person who runs it* (beginning with that job's JID).

For each job it reports, PSTAT output gives a one-word status code (such as RUN or DEPEND), supposed to reveal if the job is underway or why it is not. The meaning of some status codes (e.g., ELIG), however, is obscure or ambiguous. See the LCRM (DPCS) Reference Manual (URL: http://www.llnl.gov/LCdocs/dpcs/index.jsp?show=s4.1.2) for an explanatory, alphabetical list of the 3 dozen job-status possibilities. See also the comments on Moab job states under -A below.

SCF WARNING: On the secure network (only), LCRM divides machines into two disjoint scheduling domains: those that manage their jobs with LoadLeveler and those that manage their jobs with SLURM (see Step4 (page 28) above for details). Even if you invoke the -n or -A options (below), PSTAT only reports on those jobs within the LCRM domain to which the machine where you run it belongs. You must run PSTAT on an SCF machine in the other domain to report on any jobs in the other domain.

If you need additional, nondefault information (for example, about jobs other than your own) the following PSTAT options are especially helpful:

-n *jid*	reports a one-line status summary for any not-finished batch job (in this LCRM domain; see the SCF warning above) whose unique identifier is *jid*, regardless of who submitted that job.
-T -n *jid*	same as -n *jid* but adding -T enables reporting instead on already terminated batch jobs for 5 days after they have ended ("terminated" jobs were either removed by running PRM (page 46) or they successfully ran to completion). Using -T alone produces no output.
-A	("all") reports a one-line status summary for every not-finished batch job currently under LCRM control (in this LCRM domain; see the SCF warning above). Thus you can easily see what other jobs are in the batch system and what is the current status of each compared to your job(s). On LC machines where Moab has replaced LCRM, PSTAT with -A is fully emulated. You may also use the native Moab tool SHOWQ for a very similar comprehensive report (but some state names will differ). Moab does not enforce many LCRM resource limits on jobs, and so PSTAT run on a Moab-scheduled machine will never report that any jobs are in states that reflect those unenforced lmits (such as TOOLONG or JRESLIM). See the "Job Status Comparison" section (URL: http://www.llnl.gov/LCdocs/moab/index.jsp?show=s2.4) of the Moab at LC manual for details on how LCRM job states map into Moab job states.

-f ("full") gives a detailed, 34-property report on your current batch job, or, if used along with -n *jid*, on any batch job whose *jid* you specify, regardless of who submitted it. This reveals nonchanging job properties (such as executing host, bank, machine constraint, and node pool) as well as changing ones (such as CPU time charged, elapsed run time, and largest process size so far). Some properties reflect the whole job and some are reported per CPU. In August, 2005, the optional "project" label (-prj) replaced the former "account" attribute in all -f reports. For a field-by-field explanation of the -f report, see the Run Properties (URL: http://www.llnl.gov/LCdocs/dpcs/index.jsp?show=s4.2b) section of the LCRM (DPCS) Reference Manual. For instructions on customizing your report by using PSTAT's -o option (or the PSTAT_CONFIG environment variable), see the Reporting Memory (URL: http://www.llnl.gov/LCdocs/dpcs/index.jsp?show=s4.7) section of that manual.

On LC machines where Moab has replaced LCRM, PSTAT with -f is fully emulated. You may also use the native Moab tool CHECKJOB *jid* for a similarly detailed report on the specified job (including even its allocated nodes), but in a very different format. Moab does not enforce many LCRM resource limits on jobs, and so PSTAT run on a Moab-scheduled machine will never report that any jobs are in states that reflect those unenforced lmits (such as TOOLONG or JRESLIM). See the "Job Status Comparison" section (URL: http://www.llnl.gov/LCdocs/moab/index.jsp?show=s2.4) of the Moab at LC manual for details on how LCRM job states map into Moab job states.

-D includes DELAYED status jobs. PSTAT covers DELAYED jobs in its -n or -A reports but omits them from its default reports (on *your* jobs) unless you invoke -D.

-M -n *jid* | -m *hostname*

("machine" options) By default, PSTAT reports MULTIPLE as your job's status if it could run on any of several clustered machines with perhaps a different status on each. (See "Interpretation WARNINGS" (URL: http://www.llnl.gov/LCdocs/dpcs/index.jsp?show=s4.1.1) in the LCRM (DPCS) Reference Manual for more details.)
In either case, you can use -M or -m options to disambiguate these incomplete status reports. -M yields a separate line with a separate status for each possible target machine where *jid* could run. -m yields an explicit status for the specific *hostname* you specify (e.g., -m yana25). You cannot use both -M and -m on the same PSTAT execute line.

LRMUSAGE (formerly PCSUSAGE).
Sometimes you may want a (CPU) time-used report that sums together many runs of the same job, or all of your (recent) runs on a machine or "under" the same bank. LRMUSAGE, rather than PSTAT, is the tool to use for such totaled time-used reports. You can run LRMUSAGE on any LC production machine, open or secure, and request time-used reports for yourself (the default) or for other users sharing your bank (only). LRMUSAGE options let you specify the start and end dates for each report, and even the time units (minutes, seconds, hours), but not the time of day covered (all reports start at 0:00 and end at 24:00). LRMUSAGE options sometimes interact in unexpected ways, so consult the Bank and Allocation Manual

(URL: http://www.llnl.gov/LCdocs/banks) for an explanatory comparison of the options, with examples of typical LRMUSAGE reports on total time used.

SPJSTAT.
On IBM SP (ASC) systems where job/node interaction can be especially important, a supplementary job-reporting tool called SPJSTAT is available. Unlike PSTAT, SPJSTAT (run without arguments) reports for every current LCRM job the number of nodes that the job uses, the job's user name (e.g., gsmith), and the job's node pool (e.g., pbatch). Unfortunately, SPJSTAT reports jobs not by the (script) name that PSUB assigns them and PSTAT reports, but by their "native" job-control identifier (such as 46929, rather too mysterious to be helpful). See the POE User Guide (URL: http://www.llnl.gov/LCdocs/poe) for more details. On LC machines where Moab has replaced LCRM at the batch-job scheduler (including Linux/CHAOS machines), a native Moab tool called MJSTAT yields a job/node report very similar to SPJSTAT's.

SQUEUE.
On LC Linux (CHAOS) clusters (open and secure), SLURM (Simple Linux Utility for Resource Management) is the low-level (underneath LCRM) system that submits parallel jobs (with SRUN), allocates nodes and other resources, and tracks resource use until jobs complete. SLURM's SQUEUE tool by default reports a table of current SRUN-submitted local jobs, showing their SLURM ID (needed for SCANCEL), job and user name, time used, status, and the specific nodes committed to each. You can also customize SQUEUE reports, using a special syntax, to show your choice of any of 24 different features for SLURM-managed local jobs. Using SQUEUE instead of PSTAT could be convenient, or even essential, for effective job monitoring on new Linux machines (not yet managed by LCRM) or whenever you invoke SRUN from within an LCRM script. See the SQUEUE section (URL: http://www.llnl.gov/LCdocs/slurm/index.jsp?show=s4.3) of the SLURM Reference Manual for option details, output examples, and SQUEUE's job-state codes (which unfortunately differ from those reported by PSTAT). As SLURM also spreads to LC AIX systems and replaces IBM's LoadLeveler, remember that SQUEUE will then replace LLQ (and SCANCEL will replace LLCANCEL) even though the operating system is not CHAOS.

NUMA HARDWARE TOOLS.
On LC Linux/CHAOS clusters (such as Atlas or Zeus) that have NUMA hardware (that is, CPUs with "nonuniform memory access"), CHAOS supports extra user tools (TASKSET, NUMACTL, and NUMA-MAPS) that report the current memory-allocation policy, current heap and stack use, and current process/CPU bindings ("affinity") if any. All of these factors can strongly affect job performance on NUMA-hardware machines. See the "For Users of NUMA Nodes" section (URL: http://www.llnl.gov/LCdocs/chaos/index.jsp?show=s5.2) of LC's CHAOS reference manual for more background and usage advice.

GLOBUS MONITORING.
Users whose jobs are running on (or at least submitted to run on) trilab ASCI machines managed by the Globus system may want to consult the "ASCI Grid Monitoring Services" web site provided by Sandia National Laboratory to get job status and "state of health" reports on their Globus-managed jobs. Authorization is required; see the Globus User Guide (URL: http://www.llnl.gov/LCdocs/globus) for the current authorization rules.

Step 5b: Alter Job Features

If reports from PSTAT suggest that some features of your batch job are inappropriate, you can change (some of) them by running PALTER, or hold the job for further study by using PHOLD (until later released with PREL). PEXP, used to "expedite" a job, is not available to most users and so is not discussed here. (Likewise, only LCRM managers who are "expeditors," or others specifically authorized by them, can use special PALTER (or PSUB) options to expedite a job, exempt it from the usual job limits, or force its priority to a preassigned value. For instructions on these privileged uses of PALTER, see "Expediting and Exempting Jobs" in the LCRM (DPCS) Reference Manual (URL: http://www.llnl.gov/LCdocs/dpcs).)

On LC machines where Moab has replaced LCRM for batch scheduling, PLATER, PHOLD, and PREL are emulated (although each also has a native Moab counterpart, as noted in the Moab at LC (URL: http://www.llnl.gov/LCdocs/moab) user guide. PEXE, however, is not emulated and you must use the native Moab tool MJOBCTL to do its work.

Typical execute lines for these utilities are:

palter -n *jid* -f -tM *hh:mm* [-[no]standby]

changes the specified feature(s) of a specified batch job (already submitted), where

-n *jid*	selects for change the job whose LCRM identifier is *jid*, as reported by PSTAT. You can change only jobs that you submitted.
-f	requests a noninteractive change. Without -f, PALTER prompts you to confirm (by typing Y) the change you requested.
-tM *hh:mm*	changes the total-job CPU (or, for multi-node hosts, the per-CPU) time limit to *hh* hours and *mm* minutes. (Only LCRM managers can *increase* the time limit of a running job; users can decrease it. Both can increase the time limit of NONrunning jobs.) Other job features you can change in this way with PALTER options include the jobs's project name and bank (-prj, -b, but not if the job has started to run), as well as its machine constraints (-c) and requested node pool (-pool). WARNING: PALTER cannot change the per-process time limit (set with PSUB's -lt).
-[no]standby	changes a *queued* normal job to standby or vice versa, where -nostandby is the default and already running jobs cannot have their standby class changed with PALTER.

phold -n *jid* -f -l u

noninteractively (-f) makes the job with indentifier *jid* ineligible to run until you release it with PREL (its PSTAT status becomes "held"). You must include option -l u (specifies a user-level hold) even though that is the only kind of hold most users are allowed. PHOLD stops already running jobs only on machines (like the J90s) that support checkpointing. (Setting a job's priority to 0 has the same effect as using PHOLD, except that the job's age continues to advance.)

prel -n *jid* -f	noninteractively (-f) releases the job with indentifier *jid* to compete to run in the usual way after a previous hold.

Step 5c: Delete Problem Jobs

To delete (remove) a batch job that you have previously submitted, first discover its LCRM indentifier (its JID) by running PSTAT (page 42), and then execute the PRM utility by typing

```
prm -n jid -f
```

(or omit -f if you want an interactive prompt to confirm the deletion). You cannot delete another user's jobs, although coordinators and LCRM managers can delete jobs for any user or bank under their authority. If you ask to delete a job that has already started to run, LCRM first kills the job, then deletes it from the batch system (no extra option needed). If a job is deleted with PRM by anyone other than the job's owner, the owner receives an automatic, software-generated e-mail notification (into which the person running PRM can insert a brief, optional explanatory message).

On LC machines where Moab has replaced LCRM as the batch scheduler, PRM is emulated (but you can also invoke the counterpart native Moab tool MJOBCTL with -c). Within a single LC cluster (but not between clusters), you can use the SLURM utility SCANCEL to signal (and hence, if you wish, kill) any of your own SLURM-managed jobs or job steps on that cluster. See the SCANCEL section (URL: http://www.llnl.gov/LCdocs/slurm/index.jsp?show=s4.7) of the SLURM Reference Manual for options and usage examples.

You can defer (and schedule) your job deletion if you append to the above execute line option -A *"MM/DD/YYYY hh:mm"* (e.g., -A "08/15/1999 13:30" with many variations allowed) to specify a date and time for the deletion to occur, or option -gt *grace* (e.g., -gt 15m) to specify an amount of wall-clock time to elapse between when you execute PRM and when LCRM kills the job and deletes it.

Successfully deleted jobs are not subsequently reported by default by PSTAT. So you can only confirm the deletion by the absence of the job's entry in (for example) the PSTAT -A report, or by explicitly looking for an entry for the job by JID in the 5-day history file using PSTAT -T -n *jid*. To remove a job's history entry after confirmation, you can type

```
prm -T -n jid
```

Step 6: Check Your Log File

Your batch log file by default ends up in the directory from which you SUBMITTED your job, unless (as shown in the sample script (page 10) in this document) you explicitly request that it go elsewhere (such as into a child of your common home directory) by specifying a pathname by using PSUB's -o option.

If (but only if) you invoke all the preparatory features shown in the sample script, then your log file will contain a record of each command executed, as well as any error messages helpful for debugging.

When Things Go Wrong

This section describes what you can expect when several common but troublesome conditions arise while you run your job (interactive or batch) on LC machines. (Reviewing the "Plan Your Job Constraints" (page 20) section above can help you anticipate and avoid these problems.) This comparative approach clarifies the differences (in cause and outcome) between superficially similar problems.

Job Exceeds Its Time Limit

If you have specified a per-process time limit (with PSUB's -lt option) or a total-job time limit (with the -tM option), or if you rely on the default time limit on the machine where you run, then when your job exceeds any of these time limits:

- Your job is killed.
- Output not already written to files is lost. This is a good reason to use PSUB's -ro option (see the sample script (page 10)).
- Files not already stored may be purged or lost. This is a good reason to copy intermediate results to storage periodically.

Job's Specified Time Exceeds Your Allocation

Impossible under fair-share scheduling. You have no fixed CPU-time allocation to exceed. See the next item.

Job Exhausts Your Allocation

Impossible under fair-share scheduling. The fair-share approach tracks your total usage of computing resources ("decaying" your old usage using a half-life of one week) and adjusts your fair-share priority downward as your relative usage increases (many other factors affect this process too). But no matter how large your usage on one specific job, or on all jobs, you never "run out of time" absolutely. Instead, your job just competes more poorly against other jobs for additional time.

Job Aborts Before Your Script Starts

For each job that it begins on your behalf, LCRM *first* sets your ENVIRONMENT environment variable to the value BATCH, and *then* it sources (invokes) your dot files (such as .cshrc and .login). So if your dot files force ENVIRONMENT to the value INTERACTIVE, your LCRM-managed batch job may not run. The likely error message for this problem (in your job's error file) will be

```
Job aborted - pwsreg is not meant to be used
in a non-batch mode.
```

To avoid this conflict, carefully set ENVIRONMENT in your dot files only if it is not already defined:

```
if (! $?ENVIRONMENT) then setenv ENVIRONMENT INTERACTIVE
```

Job Starts Your Script, Then Quickly Dies

This usually means that your job was unable to create the log or other output files you specified (in your imbedded PSUB options at the top of your script, such as PSUB-ro (page 10)) because needed directories did not already exist.

A common mistake when using -ro is to create your error/log directories on only one machine in a cluster (e.g., THUNDER), but in a file system such as /var/tmp that is local to each node. If you then allow the job to run on several (or many) nodes in the cluster and if LCRM picks one for execution that lacks the needed directories, your job dies almost immediately (as soon as -ro is encountered). Using a child of your common home directory (shared by all nodes of all production machines) for error/log output avoids this problem. (You can also restrict the job to run only on a specific set of nodes by using PSUB's -pool option (page 31).)

Job Fails to Start

LCRM will not start your job if doing so would exceed any *local* (machine-specific) limits (such as a maximum allowed number of jobs/user) on its target machine(s). You can change some job properties with PALTER (page 45) (such as memory requested) to avoid some limits, but you can only wait out limits on total jobs allowed. Also, *global* (partition-wide) limits, especially on allowed jobs/bank, can have very subtle, far-reaching effects on when LCRM starts waiting jobs. Use PSTAT (page 42) to see why LCRM has not started your job; use any of the utilities in the "Planning" (page 20) section above to discover the many, interrelated limits that might be blocking your job. See also the "Resource Partition Limits" section of the LCRM (DPCS) Reference Manual. (URL: http://www.llnl.gov/LCdocs/dpcs)

Job Fails to Start Even if Expedited

Even if your job has been expedited, exempted from the usual constraints, or had its priority forced by an authorized LCRM manager, it will never start before the earliest begin-time that you specify with PSUB's -A option.

Job Gets DELAYED Status

Every machine has a maximum number of jobs *per user* that LCRM will actively consider for scheduling. If your submittal exceeds this per-user threshold, your job waits in DELAYED status until enough other jobs are scheduled that LCRM can actively consider it on its merits. The LRMMGR command "show global" reports this limit (among others). See the "Job Scheduling" section of the LCRM (DPCS) Reference Manual (URL: http://www.llnl.gov/LCdocs/dpcs) for details. Moab does not enforce this limit (and so PSTAT never reports DELAYED jobs) on the machines where Moab has replaced LCRM. Use CHECKJOB to see *why* IDLE Moab jobs are in that state.

(On SCF) PSUB Cannot Find Your Target Machine
(On SCF) PSTAT Cannot Find Your Job

LCRM divides the SCF production machines into two disjoint domains. All machines in one domain use LoadLeveler to manage their jobs, while all machines in the other domain use SLURM (note that PU manages its jobs with SLURM even though it uses AIX rather than CHAOS/Linux as its operating system). LCRM's tools, such as PSUB and PSTAT, only act on the single SCF domain where you execute them; they ignore the other domain. Hence, if you try to submit a job to run on any SCF LoadLeveler machine (such as with -c um) by executing PSUB on any SLURM machine, PSUB will return the error message "There is no host with the features you requested." Likewise, if you execute PSTAT in one domain looking for status information about any job in the other domain, even -A will fail to reveal it. On SCF machines, you must execute PSUB and PSTAT within the *same* domain where you expect to run and monitor your job(s); they do not work across domains.

Disclaimer

Keyword Index

To see an alphabetical list of keywords for this document, consult the next section (page 53).

Keyword	Description
entire	This entire document.
title	The name of this document.
scope	Topics covered in EZJOBCONTROL.
availability	Where these programs run.
who	Who to contact for assistance.
introduction	Role and goals of EZJOBCONTROL.
run-alternatives	Ways to run compared.
dpcs-policy	Job-scheduling policy summarized.
policy-background	Broad LC scheduling background.
bank-policy	How banks affect job scheduling.
shares-policy	How shares affect job scheduling.
batch-scripts	Annotated typical batch script.
psub-imbedded-commands	#PSUB script commands explained.
differences	Between-machine script differences.
job-steps	Steps to run a batch job.
job-chart	Chart of batch-job steps.
create-script	Create you job's script.
store-files	Store job's input files.
make-directories	Make all directories in advance.
plan	Plan for job constraints.
plim	Four limit-revealing utilities.
lrmmgr	Four limit-revealing utilities.
brlim	Four limit-revealing utilities.
sinfo	Four limit-revealing utilities.
phstat	Variable scheduling attributes reported.
bac	Bank privilege reports.
bt	Former bank time reports.
rdbsee	Former priority limitation reports.
phist	Past-job size reports.
pshare	Fair-share priority reports, rules.
constraint-summary	How jobs compete, summarized.
submit	Submit your job (PSUB).
psub	PSUB's submittal role.
psub-crucial	Crucial PSUB options explained.
psub-helpful	Helpful PSUB options (-c, -pool).
psub-parallel	Parallel-job PSUB options.
psub-misleading	Confusing PSUB options explained.
psub-examples	Typical PSUB execute lines.
environment-variables	Environment variables in job submittal.
monitor	Monitor your job (PSTAT).
pstat	PSTAT's reporting role.
alter	Alter job's features (PALTER).
palter	PALTER's job-adjustment role.
delete	Delete problem jobs (PRM).

Alphabetical List of Keywords

Keyword	Description
-------	-----------
a	The alphabetical index of keywords.
alter	Alter job's features (PALTER).
availability	Where these programs run.
bac	Bank privilege reports.
bank-policy	How banks affect job scheduling.
batch-scripts	Annotated typical batch script.
brlim	Four limit-revealing utilities.
bt	Former bank time reports.
check-log	Check your job's log file.
constraint-summary	How jobs compete, summarized.
create-script	Create you job's script.
date	The latest changes to EZJOBCONTROL.
delete	Delete problem jobs (PRM).
differences	Between-machine script differences.
dpcs-policy	Job-scheduling policy summarized.
entire	This entire document.
environment-variables	Environment variables in job submittal.
index	The structural index of keywords.
introduction	Role and goals of EZJOBCONTROL.
job-chart	Chart of batch-job steps.
job-problems	How common problems play out.
job-steps	Steps to run a batch job.
lrmmgr	Four limit-revealing utilities.
make-directories	Make all directories in advance.
monitor	Monitor your job (PSTAT).
palter	PALTER's job-adjustment role.
phist	Job size statistics.
phstat	Variable scheduling attributes reported.
plan	Plan for job constraints.
plim	Four limit-revealing utilities.
policy-background	Broad LC scheduling background.
prm	PRM job-deletion role.
pshare	Fair-share priority reports, rules.
pstat	PSTAT's reporting role.
psub	PSUB's submittal role.
psub-crucial	Crucial PSUB options explained.
psub-examples	Typical PSUB execute lines.
psub-helpful	Helpful PSUB options (-c, -pool).
psub-imbedded-commands	#PSUB script commands explained.
psub-misleading	Confusing PSUB options explained.
psub-parallel	Parallel-job PSUB options.
revisions	The complete revision history.
rdbsee	Former priority limitation reports.
run-alternatives	Ways to run compared.
scope	Topics covered in EZJOBCONTROL.
shares-policy	How shares affect job scheduling.
sinfo	Four limit-revealing utilities.
store-files	Store job's input files.
submit	Submit your job (PSUB).
title	The name of this document.
who	Who to contact for assistance.

Date and Revisions

Revision Date	Keyword Affected	Description of Change
29Aug07	run-alternatives	Linux helper processes die when job completes.
	plan	Large memory page info updated.
	delete	SCANCEL info, link added.
30Jul07	brlim	Moab ignores BRLIM job limits.
	monitor	Cross ref to Moab/LCRM job-state comparison.
	job-problems	Moab does not use DELAYED state.
20Jun07	differences	Warning about /var/tmp on diskless nodes.
	plan	Cross ref on NUMA hardware support.
	monitor	Cross ref on NUMA reporting tools.
23May07	psub-helpful	PSUB -s (MSUB -S) option added.
	environment-variables	MSUB -V, -v compared with PSUB -x.
20Mar07	introduction	Cross ref added to Moab manual.
	run-alternatives	Moab gradually replaces LCRM.
	plan	Moab tool replacements noted for PSHARE, LRMMGR, PLIM, PHSTAT.
	submit	Moab emulates some PSUB options, cross ref added on Moab env. vars.
	monitor	Moab emulates PSTAT, plus alternatives.
21Feb07	policy-background	Details, machine examples updated.
	batch-scripts	Compaq, MCR, ILX examples replaced.
	differences	Compaq, MCR, ILX examples replaced.
	job-steps	Compaq, MCR, ILX examples replaced.
22Aug06	run-alternatives	Tool and job-node availability explained.
	submit	White references removed.
12Jun06	differences	MPIRUN, LOADL details added.
	plan	BG/L MPIRUN constraints noted.
	psub-crucial	No PALTER support for -lt noted.
	psub-parallel	Option -g on AIX only.
	monitor	SQUEUE replaces LLQ under AIX.
07Mar06	bank-policy	Batch, interactive banks merged.
	shares-policy	PSTAT now reports AGUs.
	batch-scripts	Job ID now in LCRM e-mail.
	lrmmgr	Fewer configurable job features, running jobs not affected by changes.
	psub-crucial	-tM now per CPU, includes idle time.
	psub-parallel	Warns to use parallel file sys only, -g gets added BG/L role.
	monitor	PSTAT -f details revised.
	palter	Option -tM now per CPU.

19Jan06	plan	Environment variable issues noted.
	psub-helpful	-x option for env vars added.
	environment-variables	New section summarizes env var roles.
	index	New keyword for new section.
20Sep05	differences	White/views case added.
	plan	Two SCF domains noted.
	submit	Two SCF domains noted (for PSUB).
	monitor	Two SCF domains noted (for PSTAT).
	job-problems	Wrong-domain problem added (SCF).
29Aug05	introduction	Accounts gone in LCRM ver. 6.13.
	plan	UV, UP also have large memory pages.
	psub-helpful	-c limits only machines now,
		-pool added to limit node sets,
		-prj replaces -a (accounts defunct).
	psub-examples	Several updated, -pool added.
	monitor	Project replaces account in PSTAT output.
	palter	Constraint (-c), -pool alterable.
	job-problems	ENVIRONMENT case added.
04Apr05	phstat	New section on scheduling info tool.
	submit	Script now goes to LCRM control host.
	psub-parallel	New BG/L and RDMA options added.
	monitor	PSTAT -D option added.
	index	New keyword for new section.
	entire	LCRM manual title changed.
19Jan05	plan	Large-memory page constraint noted.
11Nov04	introduction	SLURM role clarified.
	policy-background	Details on load-balance scheduling.
	plan	SINFO (CHAOS) features added.
	index	SINFO keyword added.
	entire	LCRM stressed over DPCS.
12Aug04	monitor	SQUEUE job states cross referenced.
14Jul04	plim	Report details updated.
	psub-helpful	-np now CPUs/node.
	monitor	PCSUSAGE becomes LRMUSAGE.
19May04	monitor	SQUEUE role elaborated,
		cross reference added for details.
23Mar04	run-alternatives	SRUN's role elaborated.
	submit	SRUN's role elaborated.
	psub-helpful	SRUN's extra constraints noted.
11Nov03	psub-parallel	New -ln syntax and role explained.
	psub-helpful	-c details changed, -v added.
	psub-examples	Examples of new -ln added.
	job-problems	DELAYED status explained.
	lrmmgr	PCSMGR becomes LRMMGR everywhere.
	index	New LRMMGR keyword added.

29Oct03	run-alternatives	SRUN for "local batch" added.
	submit	SRUN under Linux/CHAOS noted.
	monitor	SQUEUE compared with SPJSTAT.
26Aug03	introduction	Cross ref to SLURM manual added.
27May03	introduction	DPCS officially becomes LCRM.
	differences	How to distinguish OCF from SCF.
	pcsmgr	Also known as LRMMGR now.
15Jan03	dpcs-policy	ILX replaces LX.
	bank-policy	Short production comments deleted.
	plan	All DCE and short prod. comments deleted.
	psub-helpful	Standby clarified, -[no]DFS deleted.
	psub-misleading	Option -sp deleted.
	psub-examples	All DCE/DFS comments deleted.
03Oct02	psub-helpful	New role for -c forest.
05Aug02	batch-scripts	IBM /var/tmp subdirectory noted.
	differences	IBM /var/tmp subdirectory noted.
15Apr02	plan	BRLIM and web site added.
	pshare	Two new options added.
	psub-helpful	-standby option added.
	psub-examples	Expanded, clarified cases.
	palter	-[no]standby option added.
	job-problems	Limits discussion expanded.
	index	New keyword added for BRLIM.
17Sep01	batch-scripts	Script names now up to 15 char.
	plan	PLIM, PCSMGR details updated.
	psub	PSUB can now expedite jobs.
	psub-helpful	Default constraints role noted.
	psub-parallel	Option -nettype added.
	psub-misleading	Options -lM, -p redefined.
	psub-examples	Parallel job case added.
	job-problems	Warning on role of -A option.
13Aug01	phist	PHIST utility explained.
18Jun01	batch-scripts	TMPDIR assigned on OCF.
	plan	PCSMGR details expanded.
	psub-helpful	DFS toggle default changes.
	pstat	Globus monitoring site noted.
07May01	batch-scripts	Role of /nfs/tmp dirs clarified.
13Mar01	psub-helpful	More DFS details cited.
	psub-examples	More DFS details cited.
	differences	IBM/POE env. var. roles noted.
	introduction	Cross ref to Bank manual added.
08Jan01	shares-policy	Ratio term in algorithm noted.
	batch-scripts	PCS_TMPDIR role explained.
	psub-parallel	Nonzero -ln needed on TC2K.
	psub-misleading	

		Former -p option disabled.
	palter	Cross ref to privileged options.
	prm	E-mail alert added.
	pstat	SPJSTAT role explained.
24Oct00	psub-helpful	Role of -noDFS clarified.
14Jul00	submit	Cross ref to LC GLOBUS guide added.
14Jun00	create-script	Percent char in name disallowed.
	submit	GLOBUS role noted.
	psub-parallel	New section, more options.
	psub-helpful	DFS toggle option added.
	index	New keyword for new section.
11May00	pstat	PCSUSAGE role explained.
01Mar00	entire	CRAY tools and examples deleted.
12Jan00	psub-helpful	Nonpreemption -np option noted.
22Nov99	batch-scripts	Output directory advice revised.
	create-script	Output directory advice revised.
	make-directories	Output directory advice revised.
	check-log	Output directory advice revised.
	job-problems	Output directory advice revised.
13Oct99	pstat	-f report expanded, altered.
	psub-helpful	Wall-clock limit -tW added.
	batch-scripts	Command echoing clarified.
11Aug99	run-alternatives	Gang scheduler alternative cited.
	policy-background	Gang scheduler guide linked in.
	palter	More constraints on time-limit changes.
	prm	Two more options added.
26Apr99	pstat	New -f report features.
06Nov98	psub-examples	DFS/DCE interaction noted.
02Sep98	entire	Links to DPCS Manual revalidated.
06Jul98	shares-policy	New section on fair-share policy.
	dpcs-policy	Revised, clarified, expanded.
	bt	BT, RDBSEE now obsolete.
	pshare	More details; on SCF too.
	constraint-summary	Now reflects fair-share sched.
	psub	Option syntax clarified.
	pstat	Technical details updated.
	prm	History log deletions too.
	job-problems	Fair-share, checkpoint effect noted.
18Mar98	who	DOCGUIDE cross ref. added.
	dpcs-policy	Fair-share role noted.
	plan	Many changes to reflect fair-share scheduling on LC's

		open machines (only).
	pshare	New section added.
	job-problems	Fair-share role noted.
	index	New keyword added.
02Feb98	who	SCF help e-mail added.
	run-alternatives	Faulty word corrected.
	batch-scripts	Scripts copied when submitted.
		Four allowed shells listed.
		Common -ro directory problem.
		Storage examples corrected.
	job-chart	Directories added to chart.
	make-directories	New keyword; directories emphasized.
	psub-crucial	-b option clarified.
	monitor	-T, -M options added.
	job-problems	Missing dirs problem added.
	index	New keyword added.
03Nov97	monitor	-m added for wrong-host status.
16Oct97	scope	Cross ref to DPCS manual added.
	introduction	Cross ref to DPCS manual added.
	monitor	Cross ref to DPCS manual added.
	policy-background	Cross ref to DPCS manual added.
	run-alternatives	30-min limit clarified.
	batch-scripts	Echo of PSUB_JOBID added.
	create-script	Echo of PSUB_JOBID added.
	psub-helpful	Multiple constraint warning added.
26Aug97	entire	Extensive updates, revisions throughout.
23Apr97	entire	First edition of LC EZJOBCONTROL manual.

TRG (29Aug07)